Landsat 7
Processing System (LPS)
Software Requirements Specification

April 28, 1995

GODDARD SPACE FLIGHT CENTER
GREENBELT, MARYLAND

Landsat 7
Processing System (LPS)
Software Requirements Specification

April 28, 1995

Prepared by:

Jeff Hosler Date
Software Manager
Landsat 7 Processing System
Code 563.2
Goddard Space Flight Center

Approved by:

J. B. Martin Date
Ground System Manager
Landsat 7 System
Code 502
Goddard Space Flight Center

Concurred by:

Joy Henegar Date
Project Manager
Landsat 7 Processing System
Code 563.1
Goddard Space Flight Center

R. Obenschain Date
Project Manager
Landsat 7 System
Code 430
Goddard Space Flight Center

Dennis M. Giblin Date
Chairman, Code 560 CCB
Goddard Space Flight Center

Abstract

The Landsat 7 Processing System (LPS) provides Landsat 7 data receipt and processing support to the Landsat 7 program, in conjunction with the Earth Science Mission Operations (ESMO) Project. The LPS receives raw wideband data from the Landsat 7 Ground Station, located at the EROS Data Center (EDC), processes it into Level 0R, browse and metadata files, and provides them to the Landsat Processes Distributed Active Archive Center (LP DAAC), also located at the EDC. The software requirements presented in this document are based on the information contained in the LPS Functional and Performance Specification (F&PS), the LPS System Design Specification, and the LPS Operations Concept document.

Keywords:

Landsat 7
Landsat 7 Processing System (LPS)
Landsat 7 Ground Station (LGS)
Landsat Processes Distributed Active Archive Center (LP DAAC)
Functional and Performance Specification (F&PS)
Mission Operations and Data Systems Directorate (MO&DSD)
Systems Management Policy (SMP)
Information Processing Division (IPD)

Preface

This specification contains the software requirements for the LPS. These requirements are based on an analysis of the requirements contained in the LPS Functional and Performance Specification (F&PS), the LPS System Design Specification, and the LPS Operations Concept document. This LPS software requirements specification, once baselined at/after the System Design/Software Requirements Review (SD/SRR) , will be controlled by the IPD (Code 560) configuration control board (CCB) and maintained and updated, as required, by the LPS Project.

This software requirements specification was prepared by:

Landsat 7 Processing System Project
Code 560
Goddard Space Flight Center
Greenbelt, MD 20771

GSFC
T. Grubb
J. Henegar
J. Hosler
E. Lee
R. McIntosh
K. Michael
D. Sames
R. Schweiss

SEAS
D. Alban
T. Aslam
B. Bacon
B. Boyce
D. Crehan
J. Freeman
A. Hall
M. Huang
C. Liu
S. Liu
N. Patel
S. Priest
T. Sawanobori
R. Shea
D. Specht
R. Tingley

Table of Contents

Section 1 - Introduction

1.1 Scope

This document presents the detailed software requirements analysis for the LPS software configuration components. The scope includes the functional, performance, operational, and programmatic requirements of the LPS.

The software requirements analysis is based on the LPS system design, the LPS Functional and Performance Specification (F&PS) and the operations concept, as well as the various technical studies completed by the LPS Project.

This document is part of the LPS project baseline. It takes effect upon approval by the Information Processing Division (Code 560) LPS Project Configuration Control Board (CCB). Proposed changes to this document require the same level of approval.

1.2 Applicable Documents

The following documents contain additional details regarding the LPS, the Landsat 7 System and Project, and external systems.

1.2.1 Specification Documents

These documents, provide the basis for developing the LPS software requirements presented in this specification.

1. Consultative Committee for Space Data Systems (CCSDS), Recommendation for Space Data System Standards; Advanced Orbiting Systems (AOS), Networks and Data Links: Architectural Specification, Blue Book, CCSDS 701.0-B-1, Issue 1, October 1989

2. NASA GSFC/MO&DSD, Landsat 7 Processing System (LPS) Functional and Performance Specification, Working Draft, 560-8FPS/0194, December 13, 1994.

3. Martin Marietta Astro Space (MMAS), Landsat 7 System Data Format Control Book (DFCB), Revision A, Volume 4 - Wideband Data, 23007702, December 2, 1994.

4. NASA GSFC, Interface Control Document (ICD) between the Landsat 7 Ground Station (LGS) and the Landsat 7 Processing System (LPS), September 1994.

5. NASA GSFC, Interface Control Document between the EOSDIS Core System (ECS) and the Landsat 7 System, Working Draft, 194-219-SE1-003, August 1994. [Note: includes LPS-LP DAAC interface requirements].

6. NASA GSFC, Operational Agreement between the Landsat 7 Processing System and the Mission Operations Center (MOC), 1995 **(TBD)**.

7. NASA GSFC, Interface Control Document between the Landsat 7 Processing System and the Image Analysis System (IAS), 23007630, 1994.

8. Computer Sciences Corporation, Seas System Development Methodology, July 1989.

10. NASA GSFC/MO&DSD, Landsat 7 Processing System (LPS) Operations Concept, Draft, 560-3OCD/0194, December 28, 1994.

1.2.2 Reference Documents

These documents are used as sources of additional and background information, as required, for developing the LPS software requirements.

1. GSFC/MO&DSD, Systems Management Policy, MDOD-8YMP/0485, July 1986.

2. National Aeronautics and Space Administration (NASA) Goddard Space Flight Center (GSFC) Landsat 7 System Specification, Review issue, 430-L-000-2-A, August 1994

3. NASA GSFC, Landsat 7 Ground Station (LGS) Functional and Performance Specification, Volume 1, Revision 1.0, November, 1994

4. NASA GSFC, Landsat 7 Ground Station (LGS) Operations Concept, 531-OCD-GS/Landsat 7, November, 1994

5. Martin Marietta Astro Space, Landsat 7 Image Assessment System (IAS) Operations Concept, Landsat 7 Library No. 5527, September, 1994

6. National Aeronautics and Space Administration (NASA) Landsat 7 Level 1 Requirements, Draft Issue, August 8, 1994.

7. United States Geological Survey (USGS)/National Oceanic and Atmospheric Administration (NOAA), Index to Landsat Worldwide Reference System (WRS) Landsats 1, 2, 3, and 4, 1982

8. Martin Marietta Astro Space, Landsat 7 System Program Coordinates System Standard, proposed update draft, 23007610A, August 1994.

9. NASA GSFC, LPS Software Management Plan, August 1994.

10. NASA GSFC, Landsat 7 Detailed Mission Requirements, January, 1995.

11. NASA GSFC, Landsat 7 System and Operations Concept (Pre-CCB Baseline version), 430-11-06--003-0, , October 1994.

12. SEAS (D. Crehan, R. McIntosh), WRS Scene Identification Algorithms (DRAFT), February 1, 1995.

13. Mission Operations and Data Systems Engineering, Renaissance Catalog of Building Blocks (Preliminary Draft), January 26, 1995.

1.3 Definitions

The following terms, as defined in this section, are commonly used throughout this document to describe the LPS operations concept.

1. Landsat 7 Contact Period:

The time duration between the start and end of a wideband data transmissions from the Landsat 7 spacecraft to a ground station. Figure 1-1 illustrates the Landsat 7 contact period concept.

2. Interval:

The time duration between the start and stop of an imaging operation (observation) of the Landsat 7 ETM+ instrument.

3. Subinterval:

A segment of raw wideband data interval received during a Landsat 7 contact period. Subintervals are caused by breaks in the wideband data stream due to communication dropouts and/or the inability of the spacecraft to transmit a complete observation (interval) within a single Landsat 7 contact period. The largest possible subinterval can

be as long as a full imaging interval. The smallest possible sub-interval can be as small as one full ETM+ scene with a time duration of approximately 24 seconds. Figure 1-1 illustrates the relationship between satellite on/off periods and satellite/ground contact periods.

4. Level 0R Files:

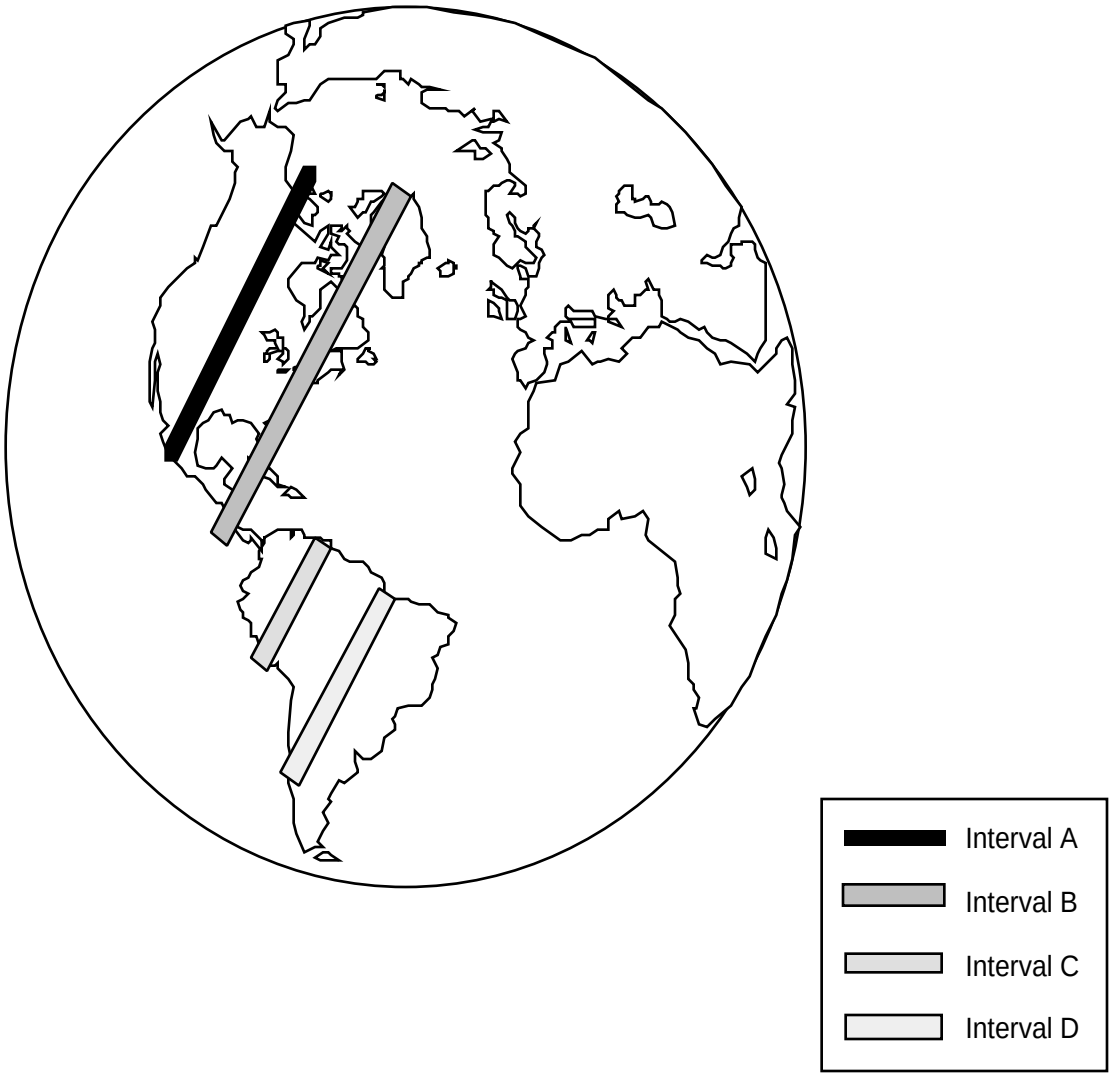
The reformatted, unrectified subinterval data having a sequence of pixels which are spatially consistent with the ground coverage and appended with radiometric calibration, attitude, and ephemeris data. Figure 1-2 illustrates the relationship of LPS files to the received subintervals.

Level 0R Instrument Data File:

Each file contains the image data from a single band in a single subinterval. The data is grouped by detectors, i.e., for a given major frame, detector 1 data is followed by detector 2 data etc. Reverse scan samples are changed to forward order. This data is nominally aligned using fixed and predetermined integer values that provide alignment for band offset, even/odd detectors, and forward and reverse scans. Quality indicators are appended for each major frame (TBR).

Calibration File:

One file is created for each subinterval. This file contains all of the calibration data received on a major frame basis for a given subinterval. This is the data received after the Scan Line Data (which follows the End of Line Code) and before the next major frame sync, as described in Applicable Document 3. The data is grouped by detectors, i.e., for a given major frame, detector 1 data is followed by detector 2 data etc. Reverse scans are reversed. The spacecraft time of the major frame corre sponding to this data is appended, as well as the status data.



Intervals Mapped to Ground Contacts

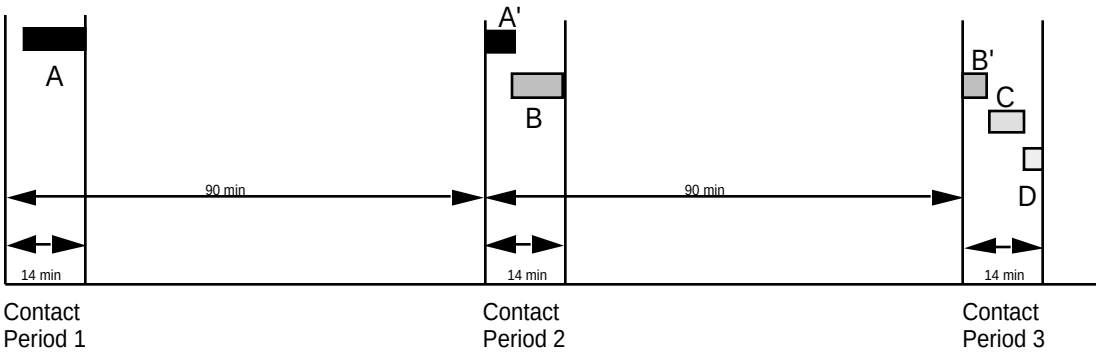


FIGURE 1-1: Landsat 7 Contact Periods Concept

LPS Wideband Data (Inputs)

LPS Files (Outputs)

a. Level 0R Files:

- Image Data
- Cal Data
- PCD
- MSCD

b. Browse Data

c. Metadata

- L0R Q&A Data

Notes:
Cal: Calibration
L0R: Level 0R
MSCD: Mirror Scan Correction Data
PCD: Payload Correction Data
Q&A: Quality and Accounting

Figure 1-2: LPS Files for Landsat 7 Contact Period 3

Mirror Scan Correction Data (MSCD):

One file is created for each subinterval. This file contains the Scan Line Data extracted from the two minor frames following the End of Line Code in each major frame of the subinterval. The Scan Line Data includes the first half scan error (FHS ERR), the second half scan error (SHS ERR), and the Scan direction (SCN DIR) information. The spacecraft time of the major frame corresponding to this data is appended.

Payload Correction Data (PCD):

One file created for each subinterval. This file contains the PCD major frames received during a subinterval on a full PCD cycle basis. Quality indicators will be appended on a minor frame basis.

5. Browse Image File:

A reduced data volume file of the Level 0R data which can be viewed to determine general ground area coverage and spatial relationships between ground area coverage and cloud coverage. This file contains reduced resolution scenes of the full resolution scene data contained in the Level 0R instrument data files of a subinterval.

Monochrome Browse:

Contains browse image data for a single band.

Multi-band Browse:

Contains browse image data from 3 predefined bands of the ETM+ Format 1 scene data.

6. Metadata:

One metadata file is created for each subinterval per string. The metadata contains information on the Level 0R data provided in the subinterval, and the names of the Level 0R instrument data, calibration data, PCD, MSCD, and browse image files associated with the subinterval. Metadata also contains quality and accounting information on the return link wideband data used in generating the Level 0R file(s). In addition, metadata includes quality and accounting information on received and processed PCD, and cloud cover assessment for the Worldwide Reference System (WRS) scene contained in the subinterval. The metadata is used to determine the subinterval and/or WRS scene level quality of the Level 0R data stored in the LP DAAC archive.

7. Return Link Quality and Accounting Data:

The data quality and accounting information collected by LPS from CCSDS AOS Grade 3 and Bose-Chaudhuri-Hocquenghem (BCH) error detection and correction processing of the raw wideband data received from LGS on a Landsat 7 contact period basis.

8. Level 0R Quality and Accounting Data:

The data quality and accounting information collected by the LPS, on a subinterval basis, from processing of the ETM+ major frames constructed from the wideband Virtual Channel Data Units (VCDUs) received during a Landsat 7 contact period.

Section 2 - Software Requirements Definition Process and Products

2.1 Process

The LPS software requirements analysis has been developed using the SEAS System Development Methodology (described in Applicable Document 8) tailored to suit the LPS project environment. The LPS software requirements analysis has been accomplished by performing the following major activities:

a. Development and analysis of an LPS software architecture that is based on LPS structured analysis, conforms to the selected hardware configuration and constraints, and maximizes the use of COTS items in its design.

b. Analysis of the LPS database which is based on the development of a conceptual and logical model of the LPS data.

c. Analysis of a user interface for the LPS, based on functional requirements and operations concepts.

d. Analysis of the LPS system requirements using a CASE tool, Cadre/Teamwork, which supports the structured analysis methodology.

e. Identification of LPS issues which, when resolved, may impact the LPS preliminary design.

2.2 Products

Several products are produced as a result of the Software Requirements analysis phase of the LPS. A model of the LPS exists in the Cadre/Teamwork CASE tool. This model includes data flow diagrams, p-specs (functional specifications), a data dictionary, and an entity relationship diagram for the LPS.

Section 3 - Reusability

3.1 Renaissance Building Blocks

There is a pool of software that has already been developed and may be reused for the LPS. By identifying candidate software components, there is the potential to save time, money, and improve an existing component (instead of developing a component).

The following is a list of candidates for reuse. Included are Renaissance Building Blocks and other MO&DSD projects. These candidates will be further investigated during the Preliminary Design Phase of the LPS.

1. The Packet Extractor/Server (building block #RT-EX-05) removes the data unit zone (preferably a CCSDS packet structure versus a mission unique structure) of a CCSDS frame. Routes selected packets and quality annotations to its clients.

2. The Telemetry VCDU Statistics (building block #RT-EX-06) keeps reception statistics on all CCSDS AOS telemetry VCDUs received. It also extracts CLCW information from real-time VCDUs and maintains this information for use by commanding building blocks.

3. The Event Logger (building block #RT-HS-01) receives event messages from all RT elements, time stamps them and logs them into a history file; provides an event server function that streams event messages to client processes based on specified filter parameters.

4. The History Logger (building block #RT-HS-02) logs annotated spacecraft telemetry frames and packets, command blocks, command echo blocks, NCC blocks, and DSN monitor blocks to the history DBM. A header record is written to the log for each data block to ensure rapid access of data during replays and browsing.

5. The History Replay (building block #RT-HS-03) replays frame and packet history files back into the real-time data path so that real-time processing of the data can be repeated. While replaying frames, the output is sent to either the Packet Extractor/Server (CCSDS missions) or directly to Telemetry

Decommutation (TDM missions). Packets are always replayed directly to Telemetry Decommutation.

6. The Generic Equation Processor (building block #RT-TM-07) uses standard algebraic and trigonometric functions, derives values from the telemetry on the data server, and places the results back on the data server.

7. The Real-Time Attitude Determination (building block #RT-TM-08) processes real-time attitude sensor telemetry data to estimate the current spacecraft attitude using a Kalman filter and/or a single frame method. The system is flexible in terms of the sensor and dynamic models used as well as which state parameters are estimated.

8. The Reports (building block #RT-US-05) generates ASCII report files using data from real-time elements. It processes page snaps, sequential prints, event dumps, telemetry frame dumps, and telemetry packet dumps to an ASCII file, laser printer, or terminal emulator window.

9. The Event Printer (building block #RT-US-06) receives events from the event logger building block in real-time and writes them to a dedicated line (events) printer.

10. The Data Distribution (building block #OL-DM-01) provides the capabilities for cataloging and distributing data products.

11. The Data Reception (building block #OL-DM-02) provides a method for receiving files from external interfaces. This software element polls a specified directory for new files at a user specified interval. When a new file is received, the poller: a) determines the file type and moves the file to a storage directory, b) updates the offline event log, c) executes a script if further processing is required.

12. The File Services (building block #OL-DM-05) provides the capability for backup storage and retrieval and archival storage, retrieval, and access control of data files. The File Server Building Block backs up active, online data file, recovers files from backup media after data loss, transfers online data files to archival storage after a predetermined amount of time and provides access to archived data files by authorized applications and users.

13. The Events Browser (building block #OL-DP-02) allows real-time and logged events information to be queried and displayed according to several different filtering criteria. A graphical "timeline" view of event occurrences by type is

provided along with the traditional scrolling text window of event messages.

14. The Data Browser and Editor (building block #OL-DP-03) provides the capability to perform formatted hexadecimal octal, decimal, and ASCII dumps of all history data (e.g. transfer frame logs, packet logs, command blocks, command echo blocks, NCC blocks, DSN monitor blocks) and level zero data sets. Dumps can be viewed on screen or can be routed to a disk or printer. Errors and missing data can be identified. Logged frames and packets can be edited for re-processing by the LZP function. Another feature of the Data Browser and Editor is that it can display the availability of history data of different types within the time range selected.

15. The TDM Processor (building block #OL-DP-06) provides counts of minor, major and incomplete major frames on a per-pass basis.

16. The Attitude Sensor Alignment and Calibration (building block #OL-SD-01) processes attitude history and attitude sensor telemetry data to estimate sensor alignment and calibration parameters using a batch filter algorithm. The system is flexible in terms of the sensor and dynamic models used as well as which state parameters are estimated.

17. The Non-Real-Time Attitude Determination (building block #OL-SD-02) processes attitude sensor telemetry data offline to estimate the spacecraft attitude at a chosen epoch. In addition, the attitude may be propagated in either direction from the chosen epoch using a user-specified data parameter. The epoch attitude may be estimated by one of several methods including batch least-squares or Kalman filter. The software is flexible in terms of the sensor and dynamic models used. Displays of uncertainties and residuals are provided so that the user may easily ascertain the validity of the solution.

18. The State Parameter Validation (building block #OL-SD-04) processes telemetry data to validate onboard computed parameters including, but not limited to, gimbal angles, spacecraft attitude, orbit information, calibration parameters, and start observations. Statistics are computed including mean difference, RMS residuals, and standard deviation.

19. The Attitude Measuring Processing (building block #OL-SD-05) processes converted telemetry data to obtain sensor and actuator measurements adjusted for misalignment and bias in a user-specified reference frame. Resulting data include, but are not limited to, magnetic field vectors, spacecraft-to-celestial

body vectors, and spacecraft body rates. the system is flexible in terms of data selection, reference frame, and output format.

20. The Session Manager (building block #OL-UI-01) provides the parameter editor capability to create, view, and modify data associated with an application. Also provides the Sequence Editor capability to sequentially execute a series of related applications.

21. The Display builder (building block #OL-OU-02) allows the construction of new Motif graphical user interfaces (GUIs). The builder takes widgets from both the Motif library and user or project-specific libraries and puts these screen objects onto a palette. From the palette these objects can be dragged to the interface that is under construction. All attributes of the screen objects can be modified before the file is saved as an industry-standard user interface language (UIL) file. Also note that the preferred method for incorporating user-defined widgets into this tool is through an industry-standard Widget Meta Language definition.

22. The Network Time Source (building block #F-TS-01) is hardware and associated device level software and other related utilities used to provide synchronization to an external timing constant. Examples are various UTC boards, NASA-36 boards, or WWV short wave boards and antenna. Provides the interface to the required external timing source.

23. The Network Time Server (building block # F-TS-02) provides the authoritative source(s) of accurate network time for all other workstations and file servers. Typically includes the machine that incorporates the Network Timer Source interface. Also serves as a relay between the Network Timer Source and all other workstations and file servers so that each of these does not require an external interface.

24. The Network Time Client (building block #F-TS-03) queries the Network Timer Server for accurate network timing that is synchronized to the Network Timer Source.

3.2 NMOS Project

1. The Distributed Process Control Program (DPCP), of the Ground Operations Technology Testbed, tool enables an operator to start a set of related processes running on one or more host computers. The sequence of steps and process dependencies is entered through use of a table that the operator sets up for the

system. The DPCP, when instructed to do so, starts all of the processes on their indicated machines in the proper order. In this way, the correct sequence of startups is consistently maintained and is executable each time the system is started. the DPCP also continues to monitor each process and presents the status information to the operator on a graphical display, showing alert status when a process has died. The DPCP allows the operator to stop all the processes with a single command. This package is completely non-intrusive in that it does not need to be compiled or linked with any of the LPS software.

3.3 SEAS Projects

1. Data transfer software can be reused from the Pacor II and DDF projects. Both of these projects perform data transfer, deal with data availability notices, and use inter-process communication to accomplish the data transfers.

2. The Distributed Application Monitor Tool (DAMT), of the Ground Operations Technology Testbed (Code 520), is helpful in analyzing performance of a software system. The DAMT may be used to monitor process on another machine, but the DAMT code must be compiled and/or linked with the application code. The DAMT is currently unable to monitor processes with the same name on different machines in the same network.

3. The user interface portion (called the Integrator) of the Centralized Information System (CIS) for the Spacelab Data Processing Facility may be reused. The Integrator is a simple display which shows the status of pipeline processes and has a window that allows the operator to easily monitor system activities. This is a C/UNIX based application that is portable.

4. The message logging portion of the Centralized Information System (CIS) for the Spacelab Data Processing Facility may be reused. This is C/UNIX based code that uses embedded SQL and is a structured and simple way of logging messages to the Integrator (mentioned above).

5. The Flight Dynamics Facility has potential software configuration items (SWCI) that have potential use within the LPS. One SWCI computes the Greenwich Hour Angle from the Julian date. Another SWCI computes the GCI sun vector.

Section 4 - Software Requirements

This section describes the LPS software requirements. The requirements in this section are intended for LPS software executing on a single string. Overall system performance requirements have been divided down to the string level.

4.1 LPS System Requirements

4.1.1 System Overview

The LPS software context level diagram (Figure 4-1) shows the interactions the LPS software has with other LPS ground system components. Raw wideband data is accepted from the LGS, the LPS generates output files, and then the LP DAAC is notified. The LPS operator issues directives to control the processing of the data. An external time source is used by five of the seven LPS subsystems to get current system time.

LPS requirements from the F&PS have been allocated to each LPS subsystem. These requirements have been further divided between operations, hardware, and software. All software allocated have been analyzed to form this software requirements specification.

There are several requirements that apply to all LPS subsystems. Refer to "Requirements Assigned All Subsystems" portion of Appendix A of the LPS System Design Specification.

4.1.2 System Functional Overview

The LPS Level 0 DFD diagram (Figure 4-2) shows the internal flow of data between the seven LPS subsystems. The raw data flows into the RDCS where it is captured. The raw data is then sent to the RDPS where all transmission artifacts are removed, leaving spacecraft mission data. This mission data (Ann_VCDU) is then passed to the MFPS. The MFPS interprets the mission data and passes payload correction data to the PCDS and image data to the IDPS. The MACS accepts accounting information from the processing subsystems and creates a metadata file. When the metadata file is created, LDTS notifies the LP DAAC that the output files are ready for transfer. LP DAAC then is responsible for transferring the files from the LPS to

the LP DAAC. Once the files have been transferred, LDTS deletes them from LPS storage.

FIGURE 4-1

LPS Software Context Level Diagram

FIGURE 4-2

LPS Level 0 Diagram

4.1.3 Open Issues

There are issues for the LPS project that remain unresolved or in need of clarification. The following issues are concerned with LPS interfaces :

- The LGS interface will be either schedule driven or data driven. The LGS has agreed to send the clock time and data only after the detection of bit sync.
- An ICD between LP-DAAC and LPS needs to be defined. It has been proposed that the LPS/LP-DAAC interface operate on a separate FDDI connected to a router. LPS is awaiting a response from LP-DAAC. Another ICD concern is the specification of DAN formats.
- The interface with the IAS needs to be stated explicitly. The data is presumed to be sent electronically. Some software may need to be written to receive this data.

The following issues concern the LPS subsystems:

- The system or subsystem responsible for the aggregation of the I and Q channels needs to be specified.
- The DFCB shows a minor frame consisting of mid scan information that needs to be extracted from the major frame. There is no functional requirement associated with this.
- The procedure for aligning calibration data is still unclear.
- Partial PCD cycles will be filled using fill values. It is in question whether this was the original intent.
- Explicit rules need to be stated regarding the majority vote decision for the PCD minor frame words.
- The following algorithms need to be identified:

  1) the PCD horizontal display shift

2) the determination method for the nearest nominal scene center.

3) the interpolation method for the WRS scene id.

- A decision is needed regarding the granularity and formatting of the LPS file outputs to the LP-DAAC in HDF.
- The precision value of the BER is in question.

The following issues concern the LPS database:

- Explicit prevention measures for data loss are still unexplained.
- Scene_Id_Setup and the Sensor_Alignment_Info are not finalized because it is not clear what information will be supplied by the IAS.
- Data will be stored for 60 days, after which it will be purged. The number of days is in question.

4.2 Programmatic Requirements

4.2.1 Development

The development environment for the LPS is the same for all SWCIs of the LPS. The goal of the LPS is to develop most functions using software (this includes application software, COTS, GOTS, system software). Doing so will reduce the need for custom hardware and facilitate future upgrades in service capability.

The application software will be developed using C and UNIX on an SGI Challenge series platform. All operating system upgrades must be coordinated with the LPS development team. All compilers and development tools must be upwardly compatible.

The Oracle DBMS COTS package will be used to manage the LPS database, generate reports (when applicable), and manage a user interface.

All government-furnished equipment will remain available to the development team during the design phases of LPS.

The LPS application software will be developed jointly by the LPS NASA developers and the LPS SEAS developers at GSFC. The operational environment will be at the EDC in Sioux Falls, South Dakota.

Since the LPS development staff will differ from the maintenance staff, the software must be developed and documented with this in mind. On-line documentation, such as LPS specific man pages will be supported.

4.2.2 Testing

The following summarizes the LPS testing approaches:

- System will be tested before delivering to EDC by LPS project personnel.
- System will be acceptance tested at EDC by EDC personnel.
- Testing at GSFC will include simulated data from GTSIM and spacecraft test data.
- LPS will support the ground system end to end testing.

4.2.3 Portability

The LPS application software needs to be upgradeable. For this reason, the application needs to be developed with portability in mind. POSIX compliant code will be written whenever possible. There may be some instances where code is not compliant due to performance requirements. These instances will be clearly documented to facilitate maintenance.

4.3 Operational Requirements

The LPS software will support and comply with the general operational requirements for the LPS system.

4.3.1 User-System Interface Requirements

The User-System Interface will be prototyped early in the preliminary design phase of the LPS. Sample screens and demonstrations will be provided to EDC personnel during that phase.

See Section 6 for more details.

4.3.2 Training

The training of the LPS operations personnel is **TBD**. Some training will be needed for both operations and maintenance personnel.

4.3.3 Maintenance

The maintenance of the LPS software will occur at the EDC by the maintenance personnel. The maintenance personnel are not planned to be the same or any subset of the development personnel.

4.4 Raw Data Capture Subsystem (RDCS)

This subsystem is responsible for receiving raw wideband data and placing it in a datastore for RDPS processing. The raw wideband data is also saved on removable media. This subsystem also restages raw wideband data from the removable media to the on-line datastore for later processing. On request, RDCS generates a data receive summary.

4.4.1 Functional Requirements

The following list summarizes the functional requirements allocated to the RDCS:

- Provide the capability to receive return link wideband data for each contact period.
- To record return link wideband data to removable media.
- To retrieve return link wideband data upon request.
- To generate an LPS wideband data receive summary for each contact period.
- Delete data for a specific contact period.

4.4.1.1 Major Functions

RDCS captures a raw data byte stream after it receives the start capture directive from the MACS and outputs this raw wideband data to a datastore for further processing by other subsystems. RDCS creates the Contact_Id information, stores it in the database, and provides it in RDC_Capture_Stat for displaying to the operator. A contact summary report is generated upon receipt of a directive from MACS which includes the data set identifier.

RDCS captures a raw data byte stream after it receives the start capture directive from the MACS and outputs this raw wideband data to a datastore for further processing by other subsystems. RDCS creates the Contact_Id information and notifies the MACS after a contact period ends. A contact summary report is generated upon receipt of a directive from MACS which includes the data set identifier.

Saving of the captured raw wideband data begins after receiving a MACS directive to start the recording to the removable media. When a request for restaging of a contact period data set is issued by the MACS, the requested data set is recovered from the removable media to the on-line store.

When a MACS directive is received requesting a data receive summary, a report is generated describing the data set. The report includes data volume and approximate number of Landsat 7 scenes along with other identifiers.

The major functions of RDCS are depicted in the following data flow diagram.

Figure 4-3
RDCS - DFD 1.0

4.4.1.2 Interface Requirements

The following two tables summarize the interface requirements for the RDCS:

Table 4.1 RDCS Interface Requirements - INPUT

Input Item Name	Source	Description
Current_Time	Time Source	System wide time source
Raw_Data_Byte_ Stream	LGS	Raw data received from LGS via LPS hardware
Configuration_Items	MACS store	LGS_Channel_Id and LPS_ Hard-ware_String_Id from the MACS datastore
RDC_Capture_ Drct	MACS	Directive to begin raw data capture
RDC_Save_Drct	MACS	Directive to begin saving raw data to removable media
RDC_Restage_ Drct	MACS	Directive to begin restaging of raw data from removable media
RDC_Delete_Drct	MACS	Directive to delete captured raw data
RDC_Rpt_Data_ Capture_Sum_Drct	MACS	Directive to begin generation of data receive summary report

Table 4.2 RDCS Interface Requirements - OUTPUT

Output Item Name	Destination	Description
Raw_WB_Sets	RDPS	Raw data to be processed
Raw_WB_Sets	Removable_ Media	Raw data saved to short-term store
RDC_Capture_Stat	MACS	Return message stating disposition of raw data capture
RDC_Save_Stat	MACS	Return message stating disposition of raw data save to removable media
RDC_Restage_ Stat	MACS	Return message stating restage of Raw_WB_Sets is complete
RDC_Delete_Stat	MACS	Return message stating completion of Raw_WB_Sets is complete
RDC_Acct	MACS	Metadata Accounting Information
Report_RDC_Data_Capture_Sum	MACS	Accounting information for a contact period

4.4.1.3 Detailed Functional Requirements

This section contains the process specifications for the lowest level processes in the RDCS data flow diagrams.

NAME:
1.1;14

TITLE:
Receive Raw Wideband Data

INPUT/OUTPUT:
Contact_Start_Time : data_inout
RDC_Capture_Stat : data_out
RDC_Acct : data_out
Raw_WB_Sets : data_out
RDC_Capture_Drct : data_in
Current_Time : data_in
Raw_Data_Byte_Stream : data_in
Configuration_Items : data_in

BODY:
Description of Process
This process captures a Raw_Data_Byte_Stream and places it in the Raw_WB_Data datastore. It also estimates the number of Megabytes received and places the estimate in the RDC_Acct datastore.

Assumptions
Preconditions
None
Postconditions
None
Constraints
None

Functional Breakdown
If RDC_Capture_Drct is "START", then
If there are three contact period data sets in the Raw_WB_Sets datastore, then
Send as RDC_Capture_Stat a message to the MACS indicating that there are already 3 contacts captured.
Read Current_Time to identify Contact_Start_Time and output to Contact_Start_Time datastore,
Start data capture of Raw_Data_Byte_Stream and place the data into the Raw_WB_Sets datastore.
Send as RDC_Capture_Stat a start capture message with Contact_Start_Time to the MACS.
Else if RDC_Capture_Drct is "STOP", then
Read Current_Time to identify Contact_Stop_Time,
Stop data capture,
Read Contact_Start_Time from the Start_Time datastore,
Calculate RDC_Acct.Rcv_Dat_Vol_Mbytes from Contact_Start_Time and Contact_Stop_Time,
Set RDC_Acct.Contact_Id.LPS_Hardware_String_Id to LPS_Configuration.LPS_Hardware_String_Id
Set RDC_Acct.Contact_Id.LGS_Channel_Id to to LPS_Configuration.LGS_Channel_Id
Set RDC_Acct.Contact_Id.Contact_Start_Time to Contact_Start_Time

Set RDC_Acct.Contact_Id.Contact_Stop_Time to Contact_Stop_Time
Send as RDC_Capture_Stat a stop capture message with Contact_Id to the MACS.

Reusability

Prototype for capturing data can be used.

NAME:
1.2;5

TITLE:
Save Raw Wideband Data

INPUT/OUTPUT:
Removable_Media : data_out
RDC_Save_Stat : data_out
RDC_Save_Drct : data_in
Raw_WB_Sets : data_in

BODY:
Description of Process
This process saves Raw_WB_Data to Removable_Media.

Assumptions
Preconditions
The current contact period Raw_WB_Data is in the Raw_WB_Sets datastore.

Postconditions
The current contact period Raw_WB_Data has been saved to the Removable_Media.
An RDC_Save_Stat has been sent to the MACS.

Constraints
None.

Functional Breakdown
Receives RDC_Save_Drct with
RDC_Save_Drct.Contact_Id.LPS_Hardware_String_Id,
RDC_Save_Drct.Contact_Id.LGS_Channel_Id,
RDC_Save_Drct.Contact_Id.Contact_Start_Time,
RDC_Save_Drct.Contact_Id.Contact_Stop_Time.
Reads Raw_WB_Data from the Raw_WB_Sets datastore.
Output the Raw_WB_Data to the Removable_Media.
Output RDC_Save_Stat to the MACS with message that saving is complete.

Reusability
The system software will be used for the majority of this function.

NAME:
1.3;7

TITLE:
Restage Raw Wideband Data

INPUT/OUTPUT:
Raw_WB_Sets : data_out
RDC_Restage_Stat : data_out
RDC_Acct : data_out
RDC_Restage_Drct : data_in
Removable_Media : data_in

BODY:
Description of Process
This process extracts Raw_WB_Data from the Removable_Media
and places it in the Raw_WB_Data datastore per Contact_Id.
It then generates the Contact_Summary and places it in the
RDC_Acct datastore and sends the RDC_Restage_Stat to the MACS.

Assumptions
Preconditions
The correct contact period Removable_Media item has been mounted.

Postconditions
The requested Raw_Data_Byte_Stream has been placed into the
Raw_WB_Sets datastore.
The accounting has been placed in the RDC_Acct datastore.
The RDC_Restage_Stat has been sent to the MACS.

Constraints
None.

Functional Breakdown
Receives RDC_Restage_Drct with
Contact_Id.LPS_Hardware_String_Id,
Contact_Id.LGS_Channel_Id,
Contact_Id.Contact_Start_Time,
Contact_Id.Contact_Stop_Time.

If requested data set is not in database then
Calculate Rcv_Dat_Vol_Mbytes from the Contact_Start_Time and the
Contact_Stop_Time.
Output Contact_Summary with
Contact_Summary.Rcv_Dat_Vol_Mbytes,
Contact_Summary.Contact_Id.LPS_Hardware_String_Id,
Contact_Summary.Contact_Id.LGS_Channel_Id,
Contact_Summary.Contact_Id.Contact_Start_Time, and
Contact_Summary.Contact_Id.Contact_Stop_Time
to the RDC_Acct datastore.
The Raw_Data_Byte_Stream is read from the Removable_Media and output to
the Raw_WB_Sets datastore for the specified RDC_Restage_Drct.
Contact_Id.
Output RDC_Restage_Stat to the MACS with message that restaging is complete.

Reusability

Most of the software used will be system software.

NAME:
1.4;12

TITLE:
Generate Data Receive Summary Report

INPUT/OUTPUT:
Report_RDC_Data_Capture_Sum : data_out
RDC_Acct : data_in
RDC_Rpt_Data_Capture_Sum_Drct : data_in

BODY:
Description of Process
This process generates the Data Receive Summary.

Assumptions
Preconditions
The RDC_Acct contains the current or requested Contact_Summary.

Postconditions
The Report_RDC_Data_Capture_Sum has been generated.

Constraints
None.

Functional Breakdown
The RDC_Rpt_Data_Capture_Sum_Drct with
Contact_Id.LPS_Hardware_String_Id,
Contact_Id.LGS_Channel_Id,
Contact_Id,Contact_Start_Time and
Contact_Id.Contact_Stop_Time is received from the MACS.
Calculate approximate number of Landsat scenes by dividing
Rcv_Dat_Vol_Mbytes by the number of megabytes per scene.
Output Report_RDC_Data_Capture_Sum to the MACS with
Report_RDC_Data_Capture_Sum.Rcv_Dat_L7_Scenes,
Report_RDC_Data_Capture_Sum.Rcv_Dat_Vol_Mbytes,
Report_RDC_Data_Capture_Sum.Contact_Id.LPS_Hardware_String_Id,
Report_RDC_Data_Capture_Sum.Contact_Id.LGS_Channel_Id,
Report_RDC_Data_Capture_Sum.Contact_Id.Contact_Start_Time and
Report_RDC_Data_Capture_Sum.Contact_Id.Contact_Stop_Time.

Reusability
This function will mostly use COTS software, ORACLE.

NAME:
1.5;4

TITLE:
Delete Raw Wideband Data

INPUT/OUTPUT:
RDC_Delete_Stat : data_out
RDC_Delete_Drct : data_in
Raw_WB_Sets : data_in

BODY:
Description of Process
This process deletes the Raw_Data_Byte_Streaam file for a specified contact period.

Assumptions
Preconditions
The Raw_Data_Byte_Stream file to be deleted exists and has been saved to Removable_Media.

Postconditions
The designated Raw_Data_Byte_Stream file has been deleted.

Constraints
None.

Functional Breakdown
Delete Raw_Data_Byte_Stream from the Raw_WB_Sets store where Raw_WB_Data.Contact_Id is equal to RDC_Delete_Drct.Contact_Id.
Output RDC_Delete_Stat to the MACS with message indicating the success or failure of the delete.

Reusability
This function will be performed by the UNIX operating system.

4.4.2 Performance Requirements

The following list summarizes the performance requirements allocated to the RDCS:

4.4.2.1 The RDCS software on each LPS string shall generate the information necessary to produce a data receive summary for received wideband data within 10 seconds of the conclusion of its capture.

4.4.2.2 The RDCS software on each LPS string shall produce a data receive summary for the most recently received wideband data within 10 seconds of the receipt of an appropriate directive from the MACS.

4.4.2.3 The RDCS software on each LPS string shall provide the capability to receive the equivalent of any combination of the format 1 and format 2 portions of 125 Landsat 7 ETM+ scenes of wideband data per day (approximately 25-30 GB per day).

4.4.2.4 The RDCS software on each LPS string shall provide the capability to copy received wideband data to removable media at a minimum rate of 7.5 Mbps.

4.4.2.5 The RDCS software on each LPS string shall provide the capability to copy received wideband data to removable media concurrently with Level 0R processing of that data.

4.4.2.6 The RDCS software on each LPS string shall provide the capability to reprocess a maximum of 10% of a string's daily input volume of wideband data (approximately 12.5 scenes or 2.5-3 GB per day).

4.4.2.7 The RDCS software on each LPS string shall provide the capability to copy received wideband data to removable media at a daily average aggregate rate of not less than 3 megabits per second (Mbps) (Includes 10% of overhead due to reprocessing).

4.4.2.8 The RDCS software on each LPS string shall maintain data processing throughput performance for all Landsat 7 raw wideband data received with a BER of one bit error in 10^5 bits, without loss of Level OR processed data and without retransmission.

4.4.2.9 The RDCS software on each LPS string shall provide the capability of receiving wideband data from a single LGS output channel at a maximum rate of 75 Mbps.

4.4.2.10 The RDCS software on each LPS string shall provide the capability to receive wideband data for Landsat 7 contact periods of up to 14 minutes.

4.4.2.11 The RDCS software on each LPS string shall output any periodic processing status information that it generates with a maximum latency of 30 seconds between outputs.

4.5 Raw Data Processing Subsystem (RDPS)

This subsystem is responsible for processing the raw wideband data on a contact period basis, including synchronization of the frames, performing various error detection and correction techniques on the data and, upon request, generating a report on the quality of the data.

4.5.1 Functional Requirements

The following list summarizes the functional requirements allocated to the RDPS:

- validate and store the processing parameters received from MACS.
- process the wideband data on a contact period basis.
- detect and synchronize on normal and inverted polarity wideband data concurrently
- the synchronization shall utilize a Search/Check/Lock/ Flywheel(SCLF) strategy.
- invert bits with inverted polarity.
- correct bit slips.
- perform PN decoding.
- perform CCSDS AOS Grade 3 service.
- store all CADUs which have failed the CCSDS AOS Grade 3 checks.
- delete VCDUs containing fill data.
- perform BCH error detection and correction.
- store all CADUs which have failed BCH error detection and correction on the mission data zone.
- generate a return link quality and accounting report on a Landsat 7 contact period basis.

- annotate the CADU with the VCID information.
- compute the Bit Error Rate (BER).

4.5.1.1 Major Functions

The Raw Data Processing Subsystem first checks the CCSDS parameters for valid values and stores these parameters if they are valid. If the CCSDS parameters are invalid default values will be used.

The raw wideband data is retrieved for the contact period requested by the operator. The frames are then synchronized using the search, check, lock, flywheel strategy. Following SCLF synchronization, the frames are aligned on byte boundaries and, if the frame sync pattern is inverted, the data is normalized reversing the data inversion. The data is then decoded to reverse the Pseudo-Random Noise encoding.

Grade 3 error detection is performed including a Cyclic Redundancy Checksum calculation and Reed-Solomon checks on the header. Any fill CADUs are discarded.

The final error detection performed on the data is the BCH Decode process that is performed on the mission and pointer data fields. This process also attempts to correct the data. Copies of the CADUs that fail the error detection processes are saved. The VCDUs are annotated to reflect the data quality, and any change in the VCID. They are also annotated to mark the end of the contact period. The bit error rate is calculated and information pertaining to the quality of the data is stored to be used in the generation of the return link quality and accounting report.

The major functions of the RDPS are depicted in the following data flow diagrams.

Figure 4-4
RDPS - DFD 2.0

Figure 4-5
RDPS - DFD 2.2

Figure 4-6
RDPS - DFD 2.3

4.5.1.2 Interface Requirements

The following two tables summarize the interface requirements for the RDPS:

Table 4.3 RDPS Interface Requirements - INPUT

Input Item Name	Source	Description
RDP_CCSDS_Parms	MACS	CCSDS Parameters
RDP_Thresholds	MACS	Error Thresholds
Raw_WB_Sets	RDCS	Raw Wideband Data Store
RDP_Process_Drct	MACS	Directive containing the contact period for which to process the raw wideband data
RDP_Rpt_Return_Link_QA_Drct	MACS	Directive containing the contact period for which to generate a return link quality and accounting report

Table 4.4 RDPS Interface Requirements - OUTPUT

Output Item Name	Destination	Description
Ann_VCDU	MFPS	VCDU annotated with contact id, data quality indicator, VCID change flag, and end of data indicator
Report_RDP_Return_ Link_QA	MACS	Return Link Quality and Accounting Report
RDP_Acct	MACS	Metadata Accounting Information
RDP_Setup_Status	MACS	Status of errors in the CCSDS parameters and/or RDP thresholds
RDP_Sync_Err_Status	MACS	Notification that synchronization errors have exceeded the error thresholds
RDP_RS_Err_Status	MACS	Notification that errors during Reed-Solomon processing have exceeded the error thresholds
RDP_CRC_Err_Status	MACS	Notification that errors during Cyclic Redundancy Check processing have exceeded the error thresholds
RDP_BCH_Err_Status	MACS	Notification that errors during BCH decode processing have exceeded the error thresholds
RDP_BER_Err_Status	MACS	Notification that the bit error rate has exceeded the BER threshold

4.5.1.3 Detailed Functional Requirements

This section contains the process specifications for the lowest level processes in the RDPS data flow diagrams.

NAME:
2.1;9

TITLE:
Validate RDP Parameters

INPUT/OUTPUT:
RDP_Thresholds : data_out
RDP_Setup_Status : data_out
RDP_CCSDS_Parms : data_out
RDP_CCSDS_Parms : data_in
RDP_Thresholds : data_in

BODY:
Description of Process

Validate the CCSDS Parameters and RDP Thresholds received from the MACS and store the valid values.

Assumptions

Preconditions

None.

Postconditions

MACS is notified if there are invalid CCSDS parameters or RDP thresholds.

Constraints

The parameters must conform to predefined thresholds
The processing of data for a contact period will not be interrupted to process updates to the CCSDS parameters.

Functional Breakdown

Validate that the RDP_CCSDS_Parms are within the following ranges:

RDP_CCSDS_Parms.CADU_Search_Tolerance: 1 to 3
RDP_CCSDS_Parms.CADU_Check_Tolerance: 0 to 3
RDP_CCSDS_Parms.CADU_Lock_Tolerance: 0 to 3
RDP_CCSDS_Parms.CADU_Flywheel_Tolerance: 0 to 3
RDP_CCSDS_Parms.CADU_Sync_Marker_Check_Error_Tolerance: 0 to 3
RDP_CCSDS_Parms.CADU_Sync_Lock_Error_Tolerance: 0 to 3
RDP_CCSDS_Parms.CADU_Bit_Slip_Correction_Extent: 0 to 3
Store the valid CCSDS parameters in Valid_CCSDS_Parms.

Check that the RDP Threshold parameters (RDP_Thresholds) are non-negative integers.

Store only the valid RDP thresholds in the Valid_RDP_Thres data store.

Send a message to the MACS specifying the names and values of any CCSDS parameters or RDP_Thresholds parameters that are in error in RDP_Status.RDP_Setup_Status.

Reusability

None.

NAME:
2.2.1

TITLE:
Perform SCLF Sync

INPUT/OUTPUT:
Chan_Acss_Acct : data_inout
Contact_Id : data_out
RDP_Sync_Err_Status : data_out
Sync_WB_Data : data_out
Valid_CCSDS_Parms : data_in
Raw_Data_Byte_Stream : data_in
RDP_Process_Drct : data_in
Sync_Thres : data_in

BODY:
Description of Process
This process uses the Search, Check, Lock, Flywheel strategy to locate frame sync patterns in the Raw_Data_Byte_Stream.

Assumptions
Preconditions
Valid_CCSDS_Parms and the RDP_Process_Drct have been received.

Postconditions
Sync_WB_Data which contains the Contact_Id, Raw_Data_Byte_Stream, frame sync marker locations, and Sync_Annotation have been sent to Align Bytes.
MACS has been notified if the cumulative number of frame sync errors have exceeded the tolerance stored in Valid_RDP_Thres.Sync_Thres
The RDP_Acct contains NULL fields for the specified Contact_Id for all return link QA information not provided by this process

Constraints
The expected frame sync pattern (1acffc1d) must exist in the data stream and must be spaced a CADU length apart, within bit slip or flywheel tolerance range.

Functional Breakdown
Retrieve the Raw_Data_Byte_Stream associated with the Contact_Id specified in the RDP_Process_Drct from the Raw_WB_Sets
Place the Contact_Id into the Sync_WB_Data.Contact_Id.
Retrieve the Valid_CCSDS_Parms.
Utilize the search/check/lock/flywheel strategy to locate the valid normal and inverted frame synchronization markers Sync_WB_Data. Polarity_Unknown_Sync.Sync and Sync_WB_Data.Polarity_Unknown_Sync. Inverted_Sync.
Detect Sync_WB_Data.Polarity_Unknown_Sync.Sync and Sync_WB_Data. Polarity_Unknown_Sync.Inverted_Sync with +- 3 bit slips, and fill or truncate the frame length accordingly
After each valid synchronization pattern is located, provide the

synchronization marker exact location in Sync_WB_Data.Sync_Loc along with the Sync_WB_Data.Sync_Annotation.
Set the Ann_VCDU.End_Of_Contact_Flag (set to TRUE for the final BCH_Chkd_VCDU for the contact period) to indicate the end of the contact period.

During all modes allow the error tolerance indicated in Valid_CCSDS_Parms.CADU_Sync_Lock_Error_Tolerance, Valid_CCSDS_Parms.CADU_Sync_Marker_Check_Error_Tolerance., Valid_CCSDS_Parms.CADU_Bit_Slip_Correction_Extent

During all modes accumulate Chan_Acss_Acct.CADU_Polarity, Chan_Acss_Acct.CADU_Bit_Slip, Chan_Acss_Acct.CADU_Sync_Error_Count, Chan_Acss_Acct.CADU_Rcv_Count, Chan_Acss_Acct.CADU_Flywheel_Count, and Chan_Acss_Acct.CADU_Missing_Count

Insert the Valid_CCSDS_Parms into the Chan_Acss_Acct.Valid_CCSDS_Parms.

Place the RDP_Process_Drct.Contact_Id into Chan_Acss_Acct.Contact_Id.

Place the Chan_Acss_Acct into the RDP_Acct store identified by the Chan_Acss_Acct.Contact_Id.

For every cumulative error detected, compare with the threshold value stored in Valid_RDP_Thres.Sync_Thres.

If the number of cumulative errors per contact period exceeds the threshold, then
Send a RDP_Status.RDP_Sync_Err_Status to the MACS indicating the tolerance has been exceeded.

Output the RDP_Process_Drct.Contact_Id

Output the Sync_WB_Data.

Reusability

A frame sync lookup table would allow bit flips in addition to bit slips.
A prototype exists for portions of this process which will be examined for reuse.

NAME:
2.2.2;3

TITLE:
Align Bytes

INPUT/OUTPUT:
Aln_CADU : data_out
Aln_Inver_CADU : data_out
Sync_WB_Data : data_in

BODY:
Description of Process
This process takes each frame of data starting with the first frame sync pattern detected and aligns the entire frame on a byte boundary.

Assumptions
Preconditions
The location of the frame sync patterns is known. The frame size is known. The raw wideband data has been extracted on a contact period basis.

Postconditions
Every CADU frame has been aligned on a byte boundary. Fill data is used if the frame size is not an even multiple of the increment being shifted.

Constraints
None

Functional Breakdown
Obtain the location of the frame sync pattern from the Sync_WB_Data.
Calculate the distance needed to shift the data.
Shift the entire frame to align on a byte boundary
Complete the Aln_CADU by adding fill data to the remainder of the frame.

If the frame sync pattern is inverted, then
Place Sync_WB_Data.Contact_Id into Aln_Inver_CADU.Contact_Id.
Output the Aln_Inver_CADU,
Otherwise
Place Sync_WB_Data.Contact_Id into Aln_CADU.Contact_Id.
Output Aln_CADU.

Reusability
A prototype exists for this process which will be examined for reuse.

NAME:
2.2.3;3

TITLE:
Deinvert Data

INPUT/OUTPUT:
Aln_CADU : data_out
Aln_Inver_CADU : data_in

BODY:
Description of Process
This process flips the bits of an entire CADU.

Assumptions
Preconditions
The frame sync pattern for the CADU is inverted.

Postconditions
The entire CADU is deinverted (normalized).

Constraints
None.

Functional Breakdown
Flip all bits of the Aln_Inver_CADU(0->1, 1->0) to create the bits of the Aln_CADU.
Place the Aln_Inver_CADU.Contact_Id into Aln_CADU.Contact_Id.

Reusability
A prototype exists for this process and it will be examined for reuse.

NAME:
2.2.4;5

TITLE:
Perform PN Decode

INPUT/OUTPUT:
Ann_CADU : data_out
Aln_CADU : data_in

BODY:
Description of Process
This process decodes the data to reverse the PN encoding on the data

Assumptions
Preconditions
CADUs have been byte aligned in the Align Bytes procoss.
Inverted data has been normalized in the Deinvert Data process.

Postconditions
Data is PN decoded

Constraints
None.

Functional Breakdown
The Aln_CADU is PN decoded using a standard pseudo-random sequence generated by a polynomial described in CCSDS 101.0-B-3, paragraphs 6.3, 6.4, and 6.5 to transform the Aln_CADU to an Ann_CADU.
Place the Aln_CADU.Contact_Id into Ann_CADU.Contact_Id.
Output the Ann_CADU.

Reusability
A prototype exists for this function which will be considered for reuse.

NAME:
2.3.1;11

TITLE:
Perform CRC Check

INPUT/OUTPUT:
CRC_Acct : data_inout
RDP_CRC_Err_Status : data_out
CRC_Failed_CADU : data_out
CRC_Chkd_CADU : data_out
Ann_CADU : data_in
CRC_Thres : data_in

BODY:
Description of Process
This process performs a 16 bit Cyclic Redundancy Checksum on the CADU

Assumptions
Preconditions
The data has been synchronized, byte aligned, CRC encoded, and PN decoded.

Postconditions
CRC errors are reported to the CRC_Acct.

Constraints
None

Functional Breakdown
Perform a Cyclic Redundancy Checksum on the Ann_CADU data
using the generator polynomial and procedure described in the CCSDS 701.0-B-1 section 5.4.9.2.1.4.2.c
Place the Ann_CADU.Contact_Id into the CRC_Acct.Contact_Id.
Update the RDP_Acct store identified by the CRC_Acct.Contact_Id with the accumulated CRC_Acct information.
If the CADU fails the CRC check, then
Place the CRC_Failed_CADU into the Faild_CADUs store identified by Ann_CADU.Contact_Id.
For every CRC error detected,
Fetch the threshold value stored in Valid_RDP_Thres.CRC_Thres.
If the cumulative number of CRC errors per contact period threshold value is exceeded, then
Send a RDP_Status.RDP_CRC_Err_Status message to the MACS indicating that the Valid_RDP_Thres.CRC_Thres tolerance has been exceeded
Output the CRC_Chkd_CADUs.

Reusability
A prototype of this process is being developed and will be examined for reuse.

NAME:
2.3.2;13

TITLE:
Perform RS_EDAC Check

INPUT/OUTPUT:
RS_Acct : data_inout
RDP_RS_Err_Status : data_out
RS_Failed_CADU : data_out
RS_Corr_CADU : data_out
CRC_Chkd_CADU : data_in
RS_Thres : data_in

BODY:
Description of Process

This process performs Reed Solomon error detection and correction on the VCDU header.

Assumptions

Preconditions

Data has been synchronized, byte aligned, PN decoded and RS encoded

Postconditions

If errors are detected and the error tolerance for Reed Solomon has not been exceeded, the VCDU header is considered corrected.

Constraints

None

Functional Breakdown

Receive CRC_Chkd_CADU and use bits 48 through 63 (16 bits) of the VCDU header as the check symbols of the shortened Reed-Solomon (10,6) code.

If the VCDU_Hdr_Bytes contained within CRC_Chkd_CADU.VCDU_Bytes are uncorrectable, then

Place the RS_Failed_CADU into the Faild_CADUs store identified by the CRC_Chkd_CADU.Contact_Id.

Accumulate the RS_Acct.VCDU_Header_Uncorrectable_Count.

Otherwise,

Place CRC_Chkd_CADU.Contact_Id into the RS_Corr_CADU.Contact_Id.

If this is a fill VCDU, then

Place the VCDU_Fill_Hdr_Bytes in the RS_Corr_CADU

Otherwise

Place the VCDU_Hdr_Bytes in the RS_Corr_CADU.

The remaining data portion of the VCDU will be identified as the RS_Corr_CADU.VCDU_Data.

Place the RS_Annotation into the RS_Corr_CADU.RS_Annotation.

If the RS_Corr_CADU has a VCID corresponding to format 1, then

Accumulate the RS_Acct.VCDU_Header_Correctable_Count.

VCDU_Hdr_Fmt1_Correctable_Err_Cnt

If the RS_Corr_CADU has a VCID corresponding to format 2, then

Accumulate the RS_Acct.VCDU_Header_Correctable_Count.

VCDU_Hdr_Fmt2_Correctable_Err_Cnt

Place the RS_Corr_CADU.Contact_Id into RS_Acct.Contact_Id.
Update the RDP_Acct store identified by the RS_Acct.Contact_Id with
the accumulated RS_Acct information.
For every uncorrectable header detected, fetch the threshold value stored
in Valid_RDP_Thres.RS_Thres.
If the cumulative number of RS errors per contact period threshold value is
exceeded, then
Send a RDP_Status.RDP_RS_Err_Status message to the MACS
stating that the Valid_RDP_Thres.RS_Thres has been exceeded.
Output the RS_Corr_CADU.

Reusability
None.

NAME:
2.3.3;3

TITLE:
Discard Fill CADUs

INPUT/OUTPUT:
Grade_3_Chkd_VCDU : data_out
RS_Corr_CADU : data_in

BODY:
Description of Process
This process discards all CADUs with fill data.

Assumptions
Preconditions
Data has been synchronized, byte aligned, and RS checked

Postconditions
Fill CADUs have been discarded

Constraints
None.

Functional Breakdown
Check the RS_Corr_CADU.VCDU_Fill_Hdr_Bytes for the reserved value of "all ones"
If the RS_Corr_CADU.VCDU_Fill_Hdr_Bytes value is all ones, discard the CADU
Otherwise,
Output the Grade_3_Chkd_VCDU.

Reusability
None.

NAME:
2.4;11

TITLE:
Decode BCH

INPUT/OUTPUT:
BCH_Acct : data_inout
RDP_BCH_Err_Status : data_out
BCH_Chkd_VCDU : data_out
BCH_Failed_VCDU : data_out
Grade_3_Chkd_VCDU : data_in
BCH_Thres : data_in

BODY:
Description of Process
This process checks the codewords of the VCDU mission and pointer data for bit errors and attempts to correct the errors.

Assumptions
Preconditions
This process will only be executed if there are any bit slips, frame sync errors, Reed-Solomon errors, or CRC errors in the Grade_3_Chkd_VCDU.

Postconditions
The VCDU has been further annotated with the mission and pointer field check quality indicators.
A copy of the VCDUs which failed the BCH EDAC on the mission data zone has been saved.

Constraints
None

Functional Breakdown
Decode the mission data zone and data pointer of the Grade_3_Chkd_VCDU which have been encoded by the polynomials defined in the Landsat 7 System Data Format Control Book, sections 3.1.2.1.2 and 3.1.2.1.3.
Attempt to correct the bit errors.
Place the Grade_3_Chkd_VCDU.Contact_Id into the BCH_Chkd_VCDU.Contact_Id.
Place the Grade_3_Chkd_VCDU data into the BCH_Chkd_VCDU
If the bits are BCH corrected, then
Insert the corrected bits into the BCH_Chkd_VCDU.
BCH_Corrected_Data
Accumulate the
BCH_Acct.BCH_Data_Corrected_Error_Count,
BCH_Acct.BCH_Pointer_Corrected_Error_Count,
BCH_Acct.BCH_Data_Uncorrected_Error_Count,
BCH_Acct.BCH_Pointer_Uncorrected_Error_Count, and
BCH_Acct.BCH_Bits_Corrected
If the error counts exceed the Valid_RDP_Thres.BCH_Thres, then
Send the RDP_Status.RDP_BCH_Err_Status containing the number of BCH errors to the MACS.

If the VCDU is uncorrectable for the mission data zone, then
Output the BCH_Failed_VCDU to the Failed_CADUs store identified by the Grade_3_Chkd_VCDU.Contact_Id.
Place the BCH_Chkd_VCDU.Contact_Id into BCH_Acct.Contact_Id.
Update the RDP_Acct store identified by the BCH_Acct.Contact_Id with the accumulated BCH_Acct information.
Place the Data_Field_Qual_Indicator into the BCH_Annotation.
Data_Field_Qual_Indicator to indicate that the data is "CORRECTABLE", "UNCORRECTABLE", or has "NO_ERRORS".
Place the BCH_Bits_Corrected into the BCH_Annotation.
Append the BCH_Annotation to the BCH_Chkd_VCDU.
Output the BCH_Chkd_VCDU.

Reusability

See the Landsat 7 Mission Data and Data Pointer BCH Decoder Prototype Description for possible reuse of the prototype.

NAME:
2.5;7

TITLE:
Annotate VCID Change

INPUT/OUTPUT:
Curr_VCID : data_out
Ann_VCDU : data_out
BCH_Chkd_VCDU : data_in
Prev_VCID : data_in
Contact_Id : data_in

BODY:
Description of Process
This process determines if there is a change in the virtual channel of the VCDU and annotates the Ann_VCDU to indicate the change.

Assumptions
Preconditions
Frame sync, CRC and Reed-Solomon checks have been performed on the VCDUs.
Fill CADUs have been discarded.
The previous VCID has been retained.

Postconditions
The VCDU has been annotated with a VCID change flag.

Constraints
None.

Functional Breakdown
Place the BCH_Chkd_VCDU.Contact_Id into the Ann_VCDU.Contact_Id.
Place the BCH_Chkd_VCDU information into the Ann_VCDU.
If the Saved_VCID.Prev_VCID differs from the Curr_VCID, then
Set the Ann_VCDU.VCID_Change_Flag to TRUE to indicate a change in the VCID.
Otherwise,
Set the Ann_VCDU.VCID_Change_Flag to FALSE to indicate no change in the VCID.
Output the Curr_VCID to the Saved_VCID data store.
Output the Ann_VCDU.

Reusability
None.

NAME:
2.6;8

TITLE:
Compute BER

INPUT/OUTPUT:
RDP_BER_Err_Status : data_out
BER_Acct : data_out
BER_Thres : data_in
RDP_Acct : data_in
Contact_Id : data_in

BODY:
Description of Process
This process computes the Bit Error Rate of the VCDU.

Assumptions
Preconditions
None.

Postconditions
None.

Constraints
None.

Functional Breakdown
Use Contact_Id to identify the associated contact period in the RDP_Acct store.
Compute the BER by dividing the sum of
RDP_Acct.BCH_Data_Corrected_Error_Count +
RDP_Acct.BCH_Data_Uncorrected_Error_Count +
RDP_Acct.BCH_Pointer_Corrected_Error_Count +
RDP_Acct.BCH_Pointer_Uncorrected_Error_Count +
RDP_Acct.CRC_Acct.CADU_CRC_Error_Count
by the RDP_Acct.CADU_Rcv_Count multiplied by the bits per CADU
If the BER exceeds the Valid_RDP_Thres.BER_Thres value, then
Send an RDP_Status.RDP_BER_Err_Status message to the MACS specifying the calculated BER.
Place the Contact_Id into the BER_Acct.Contact_Id.
Place the BER into the BER_Acct.BER.
Accumulate the BER_Acct in the RDP_Acct identified by BER_Acct.Contact_Id.

Reusability
None.

NAME:
2.7;10

TITLE:
Generate Return Link QA Report

INPUT/OUTPUT:
Report_RDP_Return_Link_QA : data_out
RDP_Acct : data_in
RDP_Rpt_Return_Link_QA_Drct : data_in

BODY:
Description of Process
This process generates a report of the raw wideband data quality and accounting information on a contact period basis

Assumptions
Preconditions
All of the required quality and accounting information will be available in the RDP account
Postconditions
None
Constraints
None

Functional Breakdown
Set Report_RDP_Return_Link_QA.Contact_Id to RDP_Rpt_Return_Link_QA_Drct.Contact_Id.
Using RDP_Rpt_Return_Link_QA_Drct.Contact_Id, extract the following information from RDP_Acct and place it into Report_RDP_Return_Link_QA:

RDP_Acct.CADU_Polarity
RDP_Acct.CADU_Bit_Slip
RDP_Acct.CADU_Sync_Error_Count
RDP_Acct.CADU_Rcv_Count
RDP_Acct.CADU_Flywheel_Count
RDP_Acct.CADU_Missing_Count
RDP_Acct.VCDU_Header_Correctable_Error_Count.
VCDU_Hdr_Fmt1_Correctable_Err_Cnt
RDP_Acct.VCDU_Header_Correctable_Error_Count.
VCDU_Hdr_Fmt2_Correctable_Err_Cnt
RDP_Acct.VCDU_Header_Uncorrectable_Error_Count
RDP_Acct.BCH_Data_Corrected_Error_Count
RDP_Acct.BCH_Pointer_Corrected_Error_Count
RDP_Acct.BCH_Data_Uncorrected_Error_Count
RDP_Acct.BCH_Pointer_Uncorrected_Error_Count
RDP_Acct.CADU_CRC_Error_Count
RDP_Acct.BER

Compute Report_RDP_Return_Link_QA.Approx_Data_Received (the approximate amount of data received in megabytes) by multiplying the RDP_Acct.CADU_Rcv_Count by the number of megabytes per CADU (.001040).

Compute Report_RDP_Return_Link_QA.Approx_Major_Frame_Count (the approximate count of major frames) by dividing the RDP_Acct.CADU_Rcv_Count by the approximate CADUs per major frame(742).

Compute Report_RDP_Return_Link_QA.Approx_ETM_Count (the approximate number of ETM+ scenes) by dividing the approximate count of major frames by the approximate frames per scene

Reusability

This function will mostly use COTS software, ORACLE.

4.5.2 Performance Requirements

The Raw Data Processing Subsystem has the following performance requirements:

4.5.2.1 The RDPS software on each LPS string shall provide the capability to process the equivalent of any combination of the format 1 and format 2 portions of 125 Landsat 7 ETM+ scenes of wideband data per day (approximately 25-30 GB per day).

4.5.2.2 The RDPS software on each LPS string shall process received wideband data at a minimum rate of not less than 7.5 Mbps. (based on a peak raw wideband throughput of 7.5 Mbps).

4.5.2.3 The RDPS software on each LPS string shall provide the capability to execute concurrently with all other LPS subsystems.

4.5.2.3.1 The RDPS software shall begin to process received raw wideband data immediately upon receipt of required inputs.

4.5.2.3.2 The RDPS software shall output the equivalent of one Landsat 7 ETM+ scene (215,445 CADUs) within 250 seconds of the time either at the beginning of processing or the time of its last output.

4.5.2.4 The RDPS software on each LPS string shall provide the capability to reprocess a maximum of 10% of a string's daily input volume of wideband data (approximately 12.5 scenes or 2.5-3 GB per day).

4.5.2.5 The RDPS software on each LPS string shall provide the capability to process received wideband data at a daily average aggregate rate of 3 megabits per second (Mbps) (Includes 10% of overhead due to reprocessing).

4.5.2.6 The RDPS software on each LPS string shall maintain data processing throughput performance for all Landsat 7 raw wideband data received with a BER of one bit error in 10^5 bits, without loss of level zero processed data and without retransmission.

4.5.2.7 The RDPS software on each LPS string shall provide the capability to retrieve retained wideband data at rates equal to or greater than 7.5 Mbps.

4.5.2.8 The RDPS software on each LPS string shall output any periodic processing status information that it generates with a maximum latency of 30 seconds between outputs.

4.6 Major Frame Processing Subsystem (MFPS)

This subsystem is responsible for identifying a major frame and determining the major frame time. This subsystem determines the subinterval from the major frame time, the VCID, or the contact period. The MFPS generates calibration and MSCD Level 0R files on a subinterval basis and scene data on a major frame basis. The MFPS generates all Level 0R quality and accounting information.

4.6.1 Functional Requirements

The following list summarizes the functional requirements allocated to the MFPS:

- validate and store the processing parameters received from MACS
- identify and collect VCDUs on a major frame basis.
- locate the minor frames that contain synchronization, major frame time, end of line code, scene data, calibration data, and MSCD.
- extract PCD/Status data and identify the number of missing VCDUs.
- determine the major frame time.
- determine the subinterval.
- deinterleave and align wideband data on a major frame basis.
- extract, process, and generate calibration and MSCD files on a subinterval basis.
- collect and generate Level 0R quality and accounting information on a subinterval basis.

4.6.1.1 Major Functions

The following is a summary of the major functions:

Setup parameters and threshold values are validated upon receipt from the MACS.

In Identify VCDUs, VCDUs are collected for one major frame according to the scan bit and the VCID. Error checking is performed on the VCDU sequence counter. Missing VCDUs are identified during this time. The VCID change and end of contact information are extracted here.

The Extract PCD process receives one VCDU at a time, plus the number of missing VCDUs, and an end of contact flag. The PCD bytes are extracted from the VCDU and sent to PCDS, along with the number of missing VCDUs, and the end of contact flag.

Parse Major Frame is responsible for locating, extracting and validating the major frame synchronization, the end of line codes, and the major frame time. Level 0R quality and accounting information is aggregated until a subinterval is detected. When a subinterval is detected, the Level 0R quality and accounting information is placed into the accounting store and the Level 0R quality and accounting aggregation store is reinitialized.

During Generate Band Data, the wide band data is deinterleaved and reversed. Band alignment takes place according to the sensor alignment information. In the case of missing major frames, MFPS outputs aligned bands containing fill.

The Extract Calibration and MSCD process handles extraction of data and the generation of the calibration and MSCD Level 0R files.

Quality and accounting from the Extract Calibration and MSCD process and Parse Major Frame process is collected on a subinterval basis and stored together. Level 0R quality and accounting reports for a given subinterval are generated on request.

The major functions of MFPS are depicted in the following data flow diagrams.

Figure 4-7
MFPS - DFD 3.0

Figure 4-8
MFPS - DFD 3.4

Figure 4-9
MFPS - DFD 3.5

Figure 4-10
MFPS - DFD 3.6

4.6.1.2 Interface Requirements

The following two tables summarize the interface requirements for the MFPS:

Table 4.5 MFPS Interface Requirements - INPUT

Input Item Name	Source	Description
Current_Time	Time Source	System wide time source
Ann_VCDU	RDPS	Annotated VCDUs
MFP_Rpt_L0R_QA_Drct	MACS	Request for a report
MFP_Parms	MACS	Setup parameters for the MFPS
MFP_Thresholds	MACS	Setup thresholds for the MFPS

Table 4.6 MFPS Interface Requirements - OUTPUT

Output Item Name	Destination	Description
PCD_Info	PCDS	PCD and related VCDU information
Major_Frame_Time	PCDS	Major frame time
Aligned_Bands	IDPS	Bands aligned along band and detector
Status_Info	IDPS	Status information from the VCDU
Report_MFP_L0R_QA	MACS	Level 0R quality and accounting information for a subinterval
MFP_Setup_Status	MACS	Return status for the MFP_Setup parameters
MFP_Cal_Status	MACS	Return status for messages
MFP_MSCD_Status	MACS	Return status for messages
MFP_Mjf_Status	MACS	Return status for messages
Sub_Intv	PCDS, IDPS, MACS	Subinterval id, start time and stop time
MFP_Acct	MACS	Metadata accounting information
Cal_File	LDTS	Calibration file
MSCD_File	LDTS	Mirror scan correction data file

4.6.1.3 Detailed Functional Requirements

This section contains the process specifications for the lowest level processes in the MFPS data flow diagrams.

NAME:
3.1;10

TITLE:
Validate MFP Parameters

INPUT/OUTPUT:
MFP_Parms : data_out
MFP_Setup_Status : data_out
MFP_Thresholds : data_out
MFP_Parms : data_in
MFP_Thresholds : data_in
Max_Alignment_Value : data_in

BODY:
Description of Process
Receive and validate all MFP_Thresholds and MFP_Parms parameters from the MACS.

Assumptions
Preconditions
None.

Postconditions
MFP_Setup_Status is output to the MACS.
All threshold values are placed into the data store Valid_MFP_Thres.
All remaining parameters are placed into the data store Valid_MFP_Parms.

Constraints
None.

Functional Breakdown
If MFP_Parms.Max_Alignment_Value exists and MFP_Parms.Max_Alignment_Value is greater than 0 and MFP_Parms.Sensor_Alignment_Info exists and some element of MFP_Parms.Sensor_Alignment_Info is greater than MFP_Parms.Max_Alignment_Value, then
Add a sensor alignment info error message to MFP_Setup_Status.
If MFP_Parms.Max_Alignment_Value does not exist or MFP_Parms.Max_Alignment_Value is less than or equal to 0 and MFP_Parms.Sensor_Alignment_Info exists and some element of MFP_Parms.Sensor_Alignment_Info is greater than Valid_MFP_Parms.Max_Alignment_Value, then
Add a sensor alignment info error message to MFP_Setup_Status.
If any of the following values are part of MFP_Parms and is less than or equal to 0
Sub_Intv_Delta
Mjf_Data_Rate
Time_Range_Tol
Part_Mnf_Tol
Maj_Vote_Tol
then
Add a sensor alignment info error message to MFP_Setup_Status.
If MFP_Status.MFP_Setup_Status contains any error messages, then

Output MFP_Status.MFP_Setup_Status.

Else

Place an acceptance message into MFP_Status.MFP_Setup_Status.
Output MFP_Status.MFP_Setup_Status.
Place the operator controls that appear in the first column
into the Valid_MFP_Parms data store.
Place the operator controls that appear in the second column
into the Valid_MFP_Thres data store.

Reusability

None.

NAME:
3.2

TITLE:
Identify VCDUs

INPUT/OUTPUT:
Ann_VCDU_Collection : data_inout
VCDU_With_Fill_Info : data_out
Major_Frame_VCDU_Set : data_out
Sub_Intv_Info : data_out
Scan_Dir_Thr : data_in
Ann_VCDU : data_in

BODY:
Description of Process
Collect Ann_VCDUs on a major frame basis and check the VCDU count.

Assumptions
Preconditions
Receive one Ann_VCDU at a time.
The Ann_VCDU.VCID_Change_Flag is set to indicate a VCID change.
The Ann_VCDU.Sync_Annotation.End_Of_Contact_Flag indicates the last VCDU of the current contact period.

Post conditions
Missing VCDUs are detected and flagged in Major_Frame_VCDU_Set and VCDU_With_Fill_Info.
One VCDU that contains the end of contact marker and information about the number of missing VCDUs is output to Extract PCD.
A set of VCDUs with the same VCID, same scan bit and same contact id is sent to Parse Major Frame. The first and last VCDU of the set may not have the same attributes as the rest.

Constraints
None.

Functional Breakdown
If the previous major frame was not processed, then
Starting with the first Ann_VCDU extract the scan bit, located in the PCD/Status field of the VCDU. This becomes the comparison variable.
Place this Ann_VCDU into the Ann_VCDU_Collection.
Read the scan bit, Ann_VCDU.VCID_Change_Flag, and Ann_VCDU. Sync_Annotation.End_Of_Contact_Flag from each successive Ann_VCDU.
If the current scan bit is different from the previous scan bit, and the Ann_VCDU.VCID_Change_Flag and Ann_VCDU.Sync_Annotation.End_Of_Contact_Flag are "false", then
Empty the Ann_VCDU_Collection store.
Increment the Ann_VCDU_Collection.Mjf_CADU_Fly_Cnt
Place the Ann_VCDU into the Ann_VCDU_Collection.
Increment Ann_VCDU_Collection.Mjf_CADU_Fly_Cnt.
If the current scan bit is different from the previous scan bit,

```
                or either the Ann_VCDU.VCID_Change_Flag, or
                Ann_VCDU.Sync_Annotation.End_Of_Contact_Flag is "true", then
                Add this Ann_VCDU as the second Ann_VCDU to the
                    Ann_VCDU_Collection store.
                Retrieve the first Ann_VCDU from the Ann_VCDU_Collection.
                Place the Ann_VCDU into VCDU_With_Fill_Info.Ann_VCDU.
                Reset VCDU_With_Fill_Info.Num_Missing_VCDUs to zero.
                Extract the VCDU counter from VCDU_With_Fill_Info.Ann_VCDU.
                    VCDU_Hdr_Bytes.
                Update the Ann_VCDU_Collection.Exp_VCDU_Ctr.
                Extract the minor frame counter from VCDU_With_Fill_Info.
                    Ann_VCDU.VCDU_Data
                Update the Ann_VCDU_Collection.Exp_Mnf_Ctr.
                Set VCDU_With_Fill_Info.Sync_Annotation.End_Of_Contact_Flag to
                    "false".
                Output VCDU_With_Fill_Info.
                Retrieve the second Ann_VCDU from the Ann_VCDU_Collection and
                    continue behind else.
        Else
            Receive Ann_VCDU.
            Extract the VCDU counter from VCDU_With_Fill_Info.Ann_VCDU.
                VCDU_Hdr_Bytes.
            Extract the minor frame counter from VCDU_With_Fill_Info.Ann_VCDU.
                VCDU_Data.
            Retrieve Ann_VCDU_Collection.Exp_VCDU_Ctr and
                Ann_VCDU_Collection.Exp_Mnf_Ctr.
            If the VCDU counter is not the same as Ann_VCDU_Collection.
                Exp_VCDU_Ctr, then
                If the minor frame counter is not the same as
                    Ann_VCDU_Collection.Exp_Mnf_Ctr, then
                        Determine the number of missing VCDUs and place
                        into Ann_VCDU_Collection.Num_Missing_VCDUs
                Else
                    Increment Ann_VCDU_Collection.VCDU_Ctr_Err by 1.
            Else If the minor frame counter is not the same as
                Ann_VCDU_Collection.Exp_Mnf_Ctr, then
                Increment Ann_VCDU_Collection.Mnf_Ctr_Err by 1.

            If the end of line code is expected in this VCDU, then
                Set Major_Frame_VCDU_Set.Exp_Eol_Ptr to point to this
                Ann_VCDU.

            Read the scan bit, Ann_VCDU.VCID_Change_Flag, and Ann_VCDU.
                Sync_Annotation.End_Of_Contact_Flag from Ann_VCDU.
            If the current scan bit is different from the previous scan bit,
                or either the Ann_VCDU.VCID_Change_Flag,
                or Ann_VCDU.Sync_Annotation.End_Of_Contact_Flag is "true",
                    then verify the Ann_VCDU_Collection.Rel_VCDU_Cnt
                    does not exceed the Valid_MFP_Thres.Scan_Dir_Thr,
                Set Major_Frame_VCDU_Set.Mjf_CADU_Rcvd_Cnt to the sum of
                    Ann_VCDU_Collection.Rel_VCDU_Cnt and
                    Ann_VCDU_Collection.Mjf_CADU_Fly_Cnt.
                Move each Ann_VCDU_Collection.Ann_VCDU,
                    and place it into Major_Frame_VCDU_Set.Ann_VCDU.
```

```
                    Place Ann_VCDU_Collection.VCDU_Ctr_Err into
                        Major_Frame_VCDU_Set.VCDU_Ctr_Err
                    Place Ann_VCDU_Collection.Mnf_Ctr_Err into
                        Major_Frame_VCDU_Set.Mnf_Ctr_Err
                    Output Major_Frame_VCDU_Set.
                    Place the Ann_VCDU.Contact_Id into Sub_Intv_Info.Contact_Id.
                    Place the Ann_VCDU.VCDU_Hdr_Bytes.VCID into Sub_Intv_Info.
                        VCID.
                    If the Ann_VCDU.VCID_Change_Flag is "true", then
                        Set the Sub_Intv_Info.VCID_Change_Flag to "true".
                    Else
                        Set the Sub_Intv_Info.VCID_Change_Flag to "false".

                    If the Ann_VCDU.Sync_Annotation.End_Of_Contact_Flag
                        is "true", then
                        Set the Sub_Intv_Info.End_Of_Contact_Flag to "true".
                    Else
                        Set the Sub_Intv_Info.End_Of_Contact_Flag to "false".
                    Output Sub_Intv_Info.
                Else
                    Add the Ann_VCDU to the Ann_VCDU_Collection.
                    Increment Ann_VCDU_Collection.Rel_VCDU_Cnt by 1.

                Place the Ann_VCDU into VCDU_With_Fill_Info.Ann_VCDU.
                Place the number of missing VCDUs into
                    VCDU_With_Fill_Info.Num_Missing_VCDUs.
                Output the VCDU_With_Fill_Info.
```

Reusability

Prototypes should be of use for the scan bit search and the check on the VCDU sequence count.

NAME:
3.3;9

TITLE:
Extract PCD

INPUT/OUTPUT:
PCD_Info : data_out
VCDU_With_Fill_Info : data_in
Sub_Intv_Id : data_in

BODY:
Description of Process
Extract PCD bytes from status field of the VCDU.

Assumptions
Preconditions
The number of missing VCDUs must be provided with VCDU_With_Fill_Info

Post conditions
PCD_Info contains the four extracted PCD bytes,
the number of missing VCDUs, and the end of contact marker.

Constraints
None.

Functional Breakdown
Extract words #1 through #4 of the PCD/Status data from the
VCDU_With_Fill_Info.Ann_VCDU.VCDU_Data zone and place into
PCD_Info.PCD_Bytes.
Place VCDU_With_Fill_Info.Num_Missing_VCDUs into
PCD_Info.Num_Missing_VCDUs.
Place VCDU_With_Fill_Info.Ann_VCDU.Sync_Annotation.
End_Of_Contact_Flag into PCD_Info.End_Of_Contact_Flag.
Place Sub_Intv_Id into PCD_Info.Sub_Intv_Id.
Output PCD_Info.

Reusability
None.

NAME:
3.4.1

TITLE:
Identify Major Frames

INPUT/OUTPUT:
Mjf_QA : data_inout
Mjf_VCDU_Data : data_out
Failed_Mjf_Data : data_out
Time_Code : data_out
Major_Frame_VCDU_Set : data_in
Maj_Vote_Tol : data_in
Sub_Intv_Info : data_in
Part_Mnf_Tol : data_in

BODY:
Description of Process
Major_Frame_VCDU_Set is searched for the major frame synchronization and the end of line code.

Assumptions
Preconditions
The set of VCDUs all have the same VCID and the same scan bit, except for the first and last VCDU.

Postconditions
Output time code minor frames.
If the major frame synchronization is not found, the set of VCDUs are placed into Failed_Mjf_data.
Processing of the Major_Frame_VCDU_Set ends.
If neither of the end of line codes are found, then
The set of VCDUs are placed into Failed_Mjf_data.
Processing of the Major_Frame_VCDU_Set ends.
If the synchronization and the end of line codes are found, then
Output Mjf_VCDU_Data.
Major frame quality and accounting information for the Major_Frame_VCDU_Set (Mjf_QA) is accumulated in the Mjf_VCDU_QA store

Constraints
None

Functional Breakdown
Search each minor frame in the first two Gap_Tagged_VCDUs.Ann_VCDU of the Major_Frame_VCDU_Set for the major frame synchronization.
Perform a majority vote on the data word groups.
The data word is accepted if it passes majority voting with a value that exceeds the Valid_MFP_Parms.Maj_Vote_Tol.
If the synchronization is not found by the end of the second Major_Frame_VCDU_Set.Gap_Tagged_VCDUs.Ann_VCDU, then
Check the partial minor frames.
If for each of the first two Major_Frame_VCDU_Set.
Gap_Tagged_VCDUs.Ann_VCDU, the

```
                                    Major_Frame_VCDU_Set.Gap_Tagged_VCDUs.
                                    Num_Missing_VCDUs is positive, then
                                    The partial minor frames of the two VCDUs will be
                                    searched for part of the major frame sync.
                                    In order for a partial minor frame to be considered,
                                    the number of data word groups must exceed
                                    Valid_MFP_Parms.Part_Mnf_Tol.
            If the synchronization is not found, then
                The Major_Frame_VCDU_Set is placed into the Failed_Mjf_Data store
                identified by Sub_Intv_Info.Contact_Id.
                Increment the Mjf_QA.Mjf_Tossed_Cnt.
                Processing for this major frame ends.
            Else
                Search each minor frame starting in the VCDU identified in
                        Major_Frame_VCDU_Set.Exp_Eol_Ptr for the end of line codes.
                Perform a majority vote on the data word groups.  The majority
                        vote count must exceed Valid_MFP_Parms.Maj_Vote_Tol in
                        order to be accepted.
                If neither end of line code is found, then
                        Begin to search again starting with the first minor frame
                        and continue until there are no more minor frames within
                        this Major_Frame_VCDU_Set.
                If either end of line code is still not found, then the
                        Major_Frame_VCDU_Set is placed into the Failed_Mjf_Data
                        store identified by Sub_Intv_Info.Contact_Id.
                        Increment the Mjf_QA.Mjf_Tossed_Cnt.
                        Increment the Mjf_QA.Mjf_Eol_Err_Cnt.
                        Processing for this major frame ends.
                Else
                        Extract the scan bit from the Status field of the VCDU.
                        Place for each Major_Frame_VCDU_Set.Gap_Tagged_VCDUs.
                        Ann_VCDU into Mjf_VCDU_Data.Gap_Tagged_VCDUs.Ann_VCDU.
                        If the scan bit indicates forward, then
                                Set Mjf_VCDU_Data.Direction_Start to point to the
                                first minor frame.
                        Else
                                Set Mjf_VCDU_Data.Direction_Start to point to the
                                last minor frame.
                        Output Mjf_VCDU_Data.
                Output the Time_Code minor frames
            Place Major_Frame_VCDU_Set.Mnf_Ctr_Err into Mjf_QA.Mnf_Ctr_Err
            Output Mjf_QA.

    Reusability
        None.
```

NAME:
3.4.2;10

TITLE:
Extract Major Frame Time

INPUT/OUTPUT:
Saved_Time : data_inout
Mjf_Time_Code_Err_Cnt : data_inout
Major_Frame_Time : data_out
Time_Code : data_in
Time_Range_Tol : data_in
Mjf_Tossed_Cnt : data_in
Current_Time : data_in
Maj_Vote_Tol : data_in
Mjf_Data_Rate : data_in

BODY:
Description of Process
The six time code minor frames are extracted and validated.

Assumptions
Preconditions
The major frame synchronization and the EOL code must be detected.

Post conditions
Major_Frame_Time is output.
Mjf_Time_Code_Err_Cnt is output.

Constraints
None.

Functional Breakdown
Receive the Time_Code.
Extract the six time code minor frames from the Time_Code.
Perform a majority vote on each data word group.
Accept the digit that exceeds the Valid_MFP_Parms.Maj_Vote_Tol.
Generate the major frame time.
Set Major_Frame_Time.Actual_Time to the major frame time.
If the sum of Saved_Time and the Mjf_Data_Rate agrees with the
Major_Frame_Time.Actual_Time, then
Set the flag Major_Frame_Time.Est_Used_Flag to "false".
Else
Determine the earliest possible major frame time by using
the Current_Time and Valid_MFP_Parms.Time_Range_Tol.
Determine if the time is anomalous.
Anomalous is defined to be a time outside the time range
between the earliest possible major frame time and
Current_Time.
If the major frame time is not anomalous, then
Set the flag Major_Frame_Time.Est_Used_Flag to "false".
Else
Accumulate the Mjf_VCDU_QA.Mjf_Time_Code_Err_Cnt.
Use Valid_MFP_Parms.Mjf_Data_Rate and Mjf_VCDU_QA.Mjf_QA.

Mjf_Tossed_Cnt to estimate the elapsed time
from the last major frame.
Add the elapsed time to Saved_Time to get the estimated
major frame time.
Place the estimated major frame time into
Major_Frame_Time.Est_Time.
Set the flag Major_Frame_Time.Est_Time_Used to "true".

If Major_Frame_Time.Est_Time_Used is "true", then
Store the Major_Frame_Time.Est_Time in the Saved_Time store.
Else
Store the Major_Frame_Time.Actual_Time in the Saved_Time store.

Output Major_Frame_Time.

Reusability
None.

NAME:
3.4.3

TITLE:
Collect VCDU Quality and Accounting

INPUT/OUTPUT:
VCDU_QA : data_inout
Major_Frame_VCDU_Set : data_in
Sub_Intv_Info : data_in
Valid_CCSDS_Parms : data_in

BODY:
Description of Process
Collect the VCDU quality and accounting data.

Assumptions
Preconditions
Quality annotation exists in the Major_Frame_VCDU_Set.Ann_VCDU.

Postconditions
The Mjf_VCDU_QA store will be updated with the newly calculated information.

Constraints
None.

Functional Breakdown
Place into VCDU_QA.Mjf_CADU_Rcvd_Cnt the Major_Frame_VCDU_Set. Mjf_CADU_Rcvd_Cnt.
Place into VCDU_QA.Mjf_CADU_Fly_Cnt the Major_Frame_VCDU_Set. Mjf_CADU_Fly_Cnt.
Place into VCDU_QA.ETM_Data_Format the Sub_Intv_Info.VCID.
Place into VCDU_QA.Mjf_CADU_Seq_Err_Cnt the Major_Frame_VCDU_Set. Mjf_CADU_Seq_Err_Cnt.
Use Sub_Intv_Info.Contact_Id to extract the RDP_Acct.Valid_CCSDS_Parms and place the information into VCDU_QA.Mjf_CADU_Sync_Info. Mjf_CADU_Sync_Strategy.Valid_CCSDS_Parms.

For each VCDU in the Major_Frame_VCDU_Set, do the following:
Place into the VCDU_QA.Mjf_CADU_Sync_Info.Mjf_CADU_Polarity with the Major_Frame_VCDU_Set.Gap_Tagged_VCDUs. Ann_VCDU.Sync_Annotation.CADU_Polarity_Flag.
Accumulate the VCDU_QA.Mjf_CADU_Sync_Info.Mjf_CADU_Bit_Slip with the Major_Frame_VCDU_Set.Gap_Tagged_VCDUs.Ann_VCDU. Sync_Annotation.CADU_Bit_Slip.
Accumulate the VCDU_QA.Mjf_CADU_Sync_Err_Cnt if the Major_Frame_VCDU_Set.Gap_Tagged_VCDUs.Ann_VCDU. Sync_Annotation.CADU_Sync_Error_Flag is "true"
Accumulate the VCDU_QA.Mjf_CADU_Missing_Cnt with the number of missing CADUs. This count should be derived from the sequence counter.
If Major_Frame_VCDU_Set.Gap_Tagged_VCDUs.Ann_VCDU.RS_Annotation is "true", then

Accumulate the VCDU_QA.Mjf_CADU_RS_Corr_Cnt.
Else
Accumulate the VCDU_QA.Mjf_CADU_RS_Uncorr_Cnt.

If the Major_Frame_VCDU_Set.Gap_Tagged_VCDUs.Ann_VCDU.
BCH_Annotation.Data_Field_Qual_Indicator is set to
"CORRECTABLE", then
Accumulate the VCDU_QA.Mjf_CADU_BCH_Corr_Cnt.
Else if the Major_Frame_VCDU_Set.Gap_Tagged_VCDUs.Ann_VCDU.
BCH_Annotation.Data_Field_Qual_Indicator set to
"UNCORRECTABLE", then
Accumulate the VCDU_QA.Mjf_CADU_BCH_Uncorr_Cnt.

Accumulate the VCDU_QA.Mjf_CADU_BCH_Bits_Corr with the
Major_Frame_VCDU_Set.Gap_Tagged_VCDUs.Ann_VCDU.
BCH_Annotation.BCH_Bits_Corrected.
Accumulate the VCDU_QA.Mjf_CADU_CRC_Err_Cnt if the
Major_Frame_VCDU_Set.Gap_Tagged_VCDUs.Ann_VCDU.
CRC_Annotation is "true".

Compute the VCDU_QA.Mjf_CADU_BER_Cnt by dividing the sum of the
VCDU_QA.Mjf_CADU_BCH_Corr_Cnt
VCDU_QA.Mjf_CADU_BCH_Uncorr_Cnt
VCDU_QA.Mjf_CADU_CRC_Err_Cnt
by the Major_Frame_VCDU_Set.Mjf_CADU_Rcvd_Cnt multiplied by the
bits per CADU

Replace the Mjf_VCDU_QA store with the new VCDU_QA information.

Reusability
None.

NAME:
3.4.4;11

TITLE:
Determine Subintervals

INPUT/OUTPUT:
Mjf_Full_Fill_Cnt : data_inout
Current_Sub_Intv_Id : data_inout
Sub_Intv_Id : data_out
Mjf_Full_Fill_Cnt : data_out
Sub_Intv : data_out
Major_Frame_Time : data_in
Sub_Intv_Delta : data_in
Sub_Intv_Info : data_in
Mjf_Data_Rate : data_in
MF_Stop_Time : data_in

BODY:
Description of Process
Determine the subinterval range.

Assumptions
Preconditions
The Valid_MFP_Parms.Sub_Intv_Delta must exist.

Postconditions
A Subinterval ID and range are generated.

Constraints
None.

Functional Breakdown
Retrieve the Valid_MFP_Parms.Sub_Intv_Delta value from the
Valid_MFP_Parms data store
If the Sub_Intv data store is empty, then
Generate a unique Sub_Intv_Id.
Place the Sub_Intv_Id into the Current_Sub_Intv_Id.

If the Sub_Intv_Info.VCID_Change_Flag is "true", then declare
the start of a new subinterval.
Place the Sub_Intv_Info.Contact_Id into Sub_Intv.Contact_Id
identified by Current_Sub_Intv_Id.
. Place the Sub_Intv_Info.VCID into Sub_Intv.VCID identified by
Current_Sub_Intv_Id.
Place the Major_Frame_Time into Sub_Intv.MF_Start_Time
identified by Current_Sub_Intv_Id.
If the Sub_Intv_Info.End_Of_Contact_Flag is "true", then declare
the end of the subinterval.
Place Major_Frame_Time into Sub_Intv.MF_Stop_Time identified by
Current_Sub_Intv_Id.
If no subinterval has yet been declared, then
Retrieve Sub_Intv.MF_Stop_Time identified by Current_Sub_Intv_Id.
Calculate a delta time value that is the difference between the

Major_Frame_Time and Sub_Intv.MF_Stop_Time.
If the delta time value exceeds Valid_MFP_Parms.Sub_Intv_Delta, or
the delta value is negative, then
Declare the start of a new subinterval.
Place the Sub_Intv_Info.Contact_Id into Sub_Intv.Contact_Id
identified by Current_Sub_Intv_Id.
. Place the Sub_Intv_Info.VCID into Sub_Intv.VCID identified by
Current_Sub_Intv_Id.
Place the Major_Frame_Time into Sub_Intv.MF_Start_Time
identified by Current_Sub_Intv_Id.
If the delta time value is less than Valid_MFP_Parms.Sub_Intv_Delta,
but positive, then no subinterval is declared.
This indicates that major frames are missing.
Calculate the number of missing major frames by dividing the
delta time value by Valid_MFP_Parms.Mjf_Data_Rate.
Place the result into Mjf_Full_Fill_Cnt.
Accumulate the Mjf_VCDU_QA.Mjf_Full_Fill_Cnt in the
Mjf_VCDU_QA store
Output Mjf_Full_Fill_Cnt.

Reusability
None.

NAME:
3.5.1;12

TITLE:
Deinterleave and Reverse Bands

INPUT/OUTPUT:
Mjf_Part_Fill_Cnt : data_out
Status_Info : data_out
Deinterleaved_Band_Data : data_out
Mjf_VCDU_Data : data_in
Fill_Value : data_in
Scan_Dir : data_in

BODY:
Description of Process
Extract the detectors of each band from the minor frames of the Mjf_VCDU_Data.

Assumptions
Preconditions
Mjf_VCDU_Data is in VCDU format.

Postconditions
The Deinterleaved_Band_Data is organized by band and detector.

Constraints
None.

Functional Breakdown
Extract the Status_Info from the PCD/Status data of the VCDU data zone.
Output the Status_Info to the IDPS.
Extract the Format_Id from the Status_Info.
Deinterleave each minor frame in the following manner according to the Status_Info.Format_Id, format 1 or format 2.
If the Scan_Dir is forward, then
Extract each byte of each minor frame starting with the minor frame location indicated by the Mjf_VCDU_Data. Direction_Start.
Place into Fmt#_Band_Data the appropriate data area according to band width and detector number.
Else
Extract each byte of each minor frame starting with the last minor frame.
The scene data minor frames will instead be deinterleaved from last to first, that is, in reverse order.
Place into Fmt#_Band_Data the appropriate data area according to band width and detector number.
If Mjf_VCDU_Data.Num_Missing_VCDUs is positive, then
Minor frames are missing.
Place Valid_MFP_Parms.Fill_Value into Deinterleaved_Band_Data for each missing minor frame.
Increment the Mjf_VCDU_QA.Mjf_Part_Full_Cnt in the Mjf_VCDU_QA store.

Output Deinterleaved_Band_Data.

Reusability
None.

NAME:
3.5.2;10

TITLE:
Align Bands

INPUT/OUTPUT:
Aligned_Bands : data_out
Deinterleaved_Band_Data : data_in
Mjf_Full_Fill_Cnt : data_in
Fill_Value : data_in
Sensor_Alignment_Info : data_in
Sub_Intv_Id : data_in

BODY:
Description of Process
Align the bands according to the Valid_MFP_Parms.Sensor_Alignment_Info.

Assumptions
Preconditions
The sensor alignment info is available.

Postconditions
The detectors of the Aligned_Bands are aligned on a major frame basis.

Constraints
The Valid_MFP_Parms.Sensor_Alignment_Info must exist.

Functional Breakdown
Place the Sub_Intv_Id into Aligned_Bands.Sub_Intv_Id.
Determine the format type from the Deinterleaved_Band_Data.Format_Id.
If Mjf_Full_Fill_Cnt is positive, then
If the format type is 1,
Fill the Aligned_Bands.Fmt1_Align_Data with the Fill_Value.
Else
Fill the Aligned_Bands.Fmt2_Align_Data with the Fill_Value.
Output Aligned_Bands for each Mjf_Full_Fill_Count.
If the format type is 1,
Align the Deinterleaved_Band_Data.Fmt1_Band_Data using the
Valid_MFP_Parms.Sensor_Alignment_Info for format 1 type data.
Place the aligned data into Aligned_Bands.Fmt1_Align_Data.
Else
Align the Deinterleaved_Band_Data.Fmt2_Band_Data using the
Valid_MFP_Parms.Sensor_Alignment_Info for format 2 type data.
Place the aligned data into Aligned_Bands.Fmt2_Align_Data.
Output Aligned_Bands.

Reusability
None.

NAME:
3.6.1;11

TITLE:
Create MSCD File

INPUT/OUTPUT:
MSCD_Data : data_inout
Sub_Intv_Id : data_out
MSCD_File_Name : data_out
MFP_MSCD_Status : data_out
MSCD_File : data_out
MF_Start_Time : data_in
File_Version_Number : data_in
Sub_Intv_Id : data_in

BODY:
Description of Process
Creates the MSCD_File_Name for the MSCD_File for each subinterval.
Stores MSCD_Data in the MSCD_File for each subinterval.

Assumptions
Preconditions
The Sub_Intv_Id is received.

Postconditions
The MSCD_File_Name and MSCD_File is created for each subinterval.
Emptys the MSCD_Data store after each subinterval.
Send MFP_Status.MFP_MSCD_Status to MACS.
The MSCD_File_Name is placed into the MFP_Acct store associated with a particular subinterval

Constraints
None.

Functional Breakdown
Create the MSCD_File_Name using LPS_Configuration.File_Version_Number and Sub_Intv.MF_Start_Time identified by Sub_Intv_Id.
Receive the MSCD_Data, which is the FHS_Err, SHS_Err, and Scan_Dir extracted from Mjf_VCDU_Data, and write the data to the MSCD_File.
Update the MFP_Acct store to contain the MSCD_File_Name for the subinterval identified by Sub_Intv_Id.
Send any MFP_Status.MFP_MSCD_Status to MACS.
Clear MSCD_Data store (to receive the next subinterval of MSCD_Data).

Reusability
This process may be used to create the Cal_File.

NAME:
3.6.2;12

TITLE:
Create Calibration File

INPUT/OUTPUT:
Cal_Data : data_inout
Sub_Intv_Id : data_out
Cal_File_Name : data_out
MFP_Cal_Status : data_out
Cal_File : data_out
MF_Start_Time : data_in
Sub_Intv_Id : data_in
File_Version_Number : data_in

BODY:
Description of Process
Creates the Cal_File_Name and the Cal_File for each subinterval.
Stores the Cal_Data in the Cal_File for each subinterval.

Assumptions
Preconditions
The Sub_Intv_Id is received.

Postconditions
The Cal_File_Name and the Cal_File is created for each subinterval.
Emptys the Cal_Data store after each subinterval.
Send MFP_Status.MFP_Cal_Status to MACS.
The Cal_File_Name is placed into the MFP_Acct store associated with a particular subinterval

Constraints
None.

Functional Breakdown
Create the Cal_File_Name using LPS_Configuration.File_Version_Number and Sub_Intv.MF_Start_Time identified by Sub_Intv_Id.
Receive the Cal_Data and write the data to the Cal_File.
Update the MFP_Acct store to contain the Cal_File_Name for the subinterval identified by Sub_Intv_Id.
Send any MFP_Status.MFP_Cal_Status to MACS.
Clear the Cal_Data store (to receive the next subinterval of Cal_Data).

Reusability
This process can be used to create the MSCD_File.

NAME:
3.6.3;10

TITLE:
Extract MSCD Data

INPUT/OUTPUT:
Scan_Dir : data_out
MSCD_Data : data_out
Mjf_VCDU_Data : data_in

BODY:
Description of Process
Extracts the MSCD_Data from the Mjf_VCDU_Data and sends the MSCD_Data to the MSCD_Data store.

Assumptions
Preconditions
The Mjf_VCDU_Data is received.

Postconditions
The MSCD_Data are extracted from the Mjf_VCDU_Data for a subinterval.
The MSCD_Data is sent to the MSCD_Data store.

Constraints
None.

Functional Breakdown
Starting after the Mjf_VCDU_Data.End_Of_Line, extract five data words from each of the first 12 groups in minor frame 1.

Extract five data words from groups 13, 14, 15, and 16 from minor frame 1.
Perform a majority vote on each data word group. This is the SHS_Err.

Extract five data words from groups 1 through 8 from minor frame 2.
Perform a majority vote on each data word group. This is the FHS_Err.

Extract five data words from groups 9 through 16 of minor frame 2.
Perform a majority vote on each data word group. This is the Scan_Dir.

If the Scan_Dir is 0 then the scan direction is reverse else if the Scan_Dir is 1 then the scan direction is forward.

Append the Scan_Dir, FHS_Err, and SHS_Err to the MSCD_Data store.
Output the Scan_Dir.

Reusability
Use the majority vote of data words from the parse major frame process to extract the MSCD_Data.

NAME:
3.6.4;12

TITLE:
Extract Calibration Data

INPUT/OUTPUT:
Cal_Data : data_out
Mjf_VCDU_Data : data_in
Fill_Value : data_in
Sensor_Alignment_Info : data_in

BODY:
Description of Process
Extracts the Cal_Data from the Mjf_VCDU_Data for each subinterval.

Assumptions
Preconditions
The Mjf_VCDU_Data is received.

Postconditions
The deinterleaved Cal_Data is stored in the Cal_Data store
for a subinterval.

Constraints
None.

Functional Breakdown
Search for the end of mirror scan correction data in Mjf_VCDU_Data.
Starting after the end of the mirror scan correction data,
Extract the Cal_Data on a minor frame basis.
Extract the scan bit.
If the scan bit is forward, then
Deinterleave the Cal_Data starting with the location indicated
by Mjf_VCDU_Data.Direction_Start
Else
Deinterleave the Cal_Data in reverse order by locating where the
start of the Cal_Data (using Mjf_VCDU_Data.Direction_Start)
to the end of the Cal_Data going backwards.
Check the Mjf_VCDU_Data.Gap_Tagged_VCDUs.Num_Missing_VCDUs
If there are missing VCDUs, then
Fill the missing VCDUs with Valid_MFP_Parms.Fill_Value
If the deinterleaved Cal_Data is format 1, then
Align the deinterleaved Cal_Data using Valid_MFP_Parms.
Sensor_Alignment_Info for format 1.
Append the deinterleaved and aligned Cal_Data to the Cal_Data store
according to band width and detector number.
Else
Align the deinterleaved Cal_Data using Valid_MFP_Parms.
Sensor_Alignment_Info for format 2.
Append the deinterleaved and aligned Cal_Data to the Cal_Data store
according to band width and detector number.

Reusability

Uses portions of the Deinterleave and Reverse Bands and Align Bands procedure from the Generate Band Data process.

NAME:
3.7;15

TITLE:
Generate Level 0R QA Report

INPUT/OUTPUT:
Report_MFP_L0R_QA : data_out
Mjf_VCDU_QA_Report_Info : data_in
MFP_Rpt_L0R_QA_Drct : data_in
MF_Start_Time : data_in
MF_Stop_Time : data_in

BODY:
Description of Process
Upon receipt of a report request, generate reports.

Assumptions
Preconditions
There must be data in MFP_Acct.Mjf_VCDU_QA.

Postconditions
A Level 0R quality and accounting report on a subinterval basis is generated.

Constraints
None.

Functional Breakdown
Use the MFP_Rpt_L0R_QA_Drct.Sub_Intv_Id to retrieve the following information:

Mjf_VCDU_QA.VCDU_QA.Mjf_Count,
Mjf_VCDU_QA.VCDU_QA.Mjf_CADU_Sync_Info,
Mjf_VCDU_QA.VCDU_QA.Mjf_CADU_Sync_Err_Cnt,
Mjf_VCDU_QA.VCDU_QA.Mjf_CADU_Rcvd_Cnt,
Mjf_VCDU_QA.VCDU_QA.Mjf_CADU_Fly_Cnt,
Mjf_VCDU_QA.VCDU_QA.Mjf_CADU_Missing_Cnt,
Mjf_VCDU_QA.VCDU_QA.Mjf_CADU_RS_Corr_Cnt,
Mjf_VCDU_QA.VCDU_QA.Mjf_CADU_RS_Uncorr_Cnt,
Mjf_VCDU_QA.VCDU_QA.Mjf_CADU_BCH_Corr_Cnt,
Mjf_VCDU_QA.VCDU_QA.Mjf_CADU_BCH_Uncorr_Cnt,
Mjf_VCDU_QA.VCDU_QA.Mjf_CADU_CRC_Err_Cnt,
Mjf_VCDU_QA.Mjf_QA.Mjf_Full_Fill_Cnt,
Mjf_VCDU_QA.Mjf_QA.Mjf_Part_Fill_Cnt,
Mjf_VCDU_QA.VCDU_QA.Mjf_CADU_BER_Cnt,

into the Report_MFP_L0R_QA.Mjf_VCDU_QA_Report_Info

Sub_Intv.MF_Start_Time, and
Sub_Intv.MF_Stop_Time

into the Report_MFP_L0R_QA.

Create the report format for the Report_MFP_L0R_QA.
Output the Report_MFP_L0R_QA.

Reusability
Report generation routines exist within Oracle.

NAME:
3.8

TITLE:
Collect Quality and Accounting

INPUT/OUTPUT:
Mjf_VCDU_QA : data_inout
Mjf_VCDU_QA : data_out
MFP_Mjf_Status : data_out
Sub_Intv_Id : data_out
Valid_MFP_Thres : data_in
Sub_Intv_Id : data_in

BODY:
Description of Process
Collect the Mjf_VCDU_QA data, check threshold values, and place the information into the accounting store on a subinterval basis.

Assumptions
Preconditions
Mjf_VCDU_QA has quality and accounting data on a major frame basis.

Postconditions
The Mjf_VCDU_QA is placed into the MFP_Acct store with NULL values for the MSCD_File_Name and Cal_File_Name on a subinterval basis.

Constraints
None.

Functional Breakdown
Upon receipt of the Sub_Intv_Id,
Extract the information out of Mjf_VCDU_QA and place into the MFP_Acct store identified by Sub_Intv_Id.
Test for the following thresholds:
If Mjf_VCDU_QA.VCDU_QA.Mjf_CADU_Seq_Err_Cnt exceeds Valid_MFP_Thres.Mjf_CADU_Seq_Err_Thr, then
Add an error message to MFP_Status.MFP_Mjf_Status.
If Mjf_VCDU_QA.Mjf_QA.Mnf_Ctr_Err exceeds Valid_MFP_Thres. Mnf_Ctr_Thr, then
Add an error message to MFP_Status.MFP_Mjf_Status.
If Mjf_VCDU_QA.VCDU_QA.Mjf_CADU_Sync_Err_Cnt exceeds Valid_MFP_Thres.Sync_Thr, then
Add an error message to MFP_Status.MFP_Mjf_Status.
If Mjf_VCDU_QA.Mjf_QA.Mjf_Eol_Err_Cnt exceeds Valid_MFP_Thres. Eol_Thr, then
Add an error message to MFP_Status.MFP_Mjf_Status.
If Mjf_VCDU_QA.Mjf_Time_Code_Err_Cnt exceeds Valid_MFP_Thres. Tc_Thr, then
Add an error message to MFP_Status.MFP_Mjf_Status.
If Mjf_VCDU_QA.Mjf_Full_Fill_Cnt exceeds Valid_MFP_Thres. Full_Mjf_Thr, then
Add an error message to MFP_Status.MFP_Mjf_Status.

If Mjf_VCDU_QA.Mjf_Part_Fill_Cnt exceeds Valid_MFP_Thres.
Part_Mjf_Thr, then
Add an error message to MFP_Status.MFP_Mjf_Status.
If any error messages are in MFP_Status.MFP_Mjf_Status, then
Output MFP_Status.MFP_Mjf_Status.
Empty the Mjf_VCDU_QA store of all data pertaining to the new subinterval.

Reusability
None.

4.6.2 Performance Requirements

The following list summarizes the performance requirements allocated to the MFPS:

4.6.2.1 The MFPS software on each LPS string shall provide the capability to process the equivalent of any combination of the format 1 and format 2 portions of 125 Landsat 7 ETM+ scenes of wideband data per day (approximately 25-30 GB per day).

4.6.2.2 The MFPS software on each LPS string shall process received data at a minimum rate of not less than 7.5 Mbps. (based on a minimum raw wideband throughput of 7.5 Mbps).

4.6.2.3 The MFPS software on each LPS string shall provide the capability to execute concurrently with all other LPS subsystems.

4.6.2.3.1 The MFPS software shall begin to process received raw data immediately upon receipt of required inputs.

4.6.2.3.2 The MFPS software shall output the equivalent of one Landsat 7 ETM+ scene worth's of Major Frames and PCD within 240 seconds of the receipt of all required inputs.

4.6.2.4 The MFPS software on each LPS string shall provide the capability to reprocess a maximum of 10% of a string's daily input volume of wideband data (approximately 12.5 scenes or 2.5-3 GB per day).

4.6.2.5 The MFPS software on each LPS string shall provide the capability to process received wideband data at a daily average aggregate rate of 3 megabits per second (Mbps) (Includes 10% of overhead due to reprocessing).

4.6.2.6 The MFPS software on each LPS string shall maintain data processing throughput performance for all Landsat 7 raw wideband data received with a BER of one bit error in 10^5 bits, without loss of level zero R processed data and without retransmission.

4.6.2.7 The MFPS software on each LPS string shall output any periodic processing status information that it generates with a maximum latency of 30 seconds between outputs.

4.7 Payload Correction Data Subsystem (PCDS)

The PCDS is responsible for building PCD major frames using the PCD information words taken from the PCD/Status section of the virtual channel data unit (VCDU). The PCD major frames contain data that is needed to geometrically correct the ETM+ imagery. In addition, the PCDS identifies all Enhanced Thematic Mapper Plus (ETM+) scenes in accordance with the Worldwide Reference System (WRS) on a subinterval basis.

4.7.1 Functional Requirements

The following list summarizes the functional requirements allocated to the PCDS:

- validate and store the processing parameters received from MACS.
- provide the capability to synchronize on PCD bytes for assembling PCD minor frames, using fill for missing PCD data. Assemble PCD major frames using the PCD minor frames.
- provide the capability to generate PCD file(s) on a subinterval basis.
- provide the capability to collect and store PCD quality and accounting and processed PCD quality and accounting data on a subinterval basis.
- provide the capability to perform ETM+ scene identification within an accuracy of 1 kilometer in accordance with the Worldwide Reference System (WRS) scheme.
- calculate the spacecraft drift time based on information available in the PCD.
- provide the capability to identify the presence of calibration door activities.
- allow the operator to select thresholds for results and errors reported by the LPS. Using the thresholds, automatically generate messages and alarms to alert the operator of LPS results and errors exceeding selected thresholds.

4.7.1.1 Major Functions

The PCDS accepts the PCD_Info from the Major Frame Processing Subsystem. PCDS extracts the PCD information words from the PCD_Info that will be used to build PCD minor frames. If any of the PCD information words are missing, then fill will be used in place of the missing information word. The PCD minor frame accounting is generated on a subinterval basis.

The PCD minor frames are used to build PCD major frames. PCD major frame quality and accounting is generated per PCD major frame. Four consecutive PCD major frames are grouped into a Full_PCD_Cycle. The Full_PCD_Cycles are used to build the PCD_File on a subinterval basis.

In addition, PCDS uses the Full_PCD_Cycles to extract the drift time for use by the Image Data Processing Subsystem (IDPS) on a subinterval basis. The PCDS identifies the ETM+ scenes using the Full_PCD_Cycles in accordance with the Worldwide Reference System (WRS) scheme, and provides the scene parameters, per scene, on a subinterval basis.

The major functions of PCDS are depicted in the following data flow diagrams.

Figure 4-11
PCDS - DFD 4.0

Figure 4-12
PCDS - DFD 4.2

Figure 4-13
PCDS - DFD 4.3

Figure 4-14
PCDS - DFD 4.4

4.7.1.2 Interface Requirements

The following two tables summarize the interface requirements for the PCDS:

Table 4.7 PCDS Interface Requirements - INPUT

Input Item Name	Source	Description
PCD_Thresholds	MACS	Thresholds used by PCDS
PCD_Parms	MACS	Parameters used by PCDS.
LPS_Configuration	MACS	String specific information.
Current_Time	Time Source	System wide time source
Major_Frame_Time	MFPS	The time associated with one major frame.
PCD_Info	MFPS	The PCD words and information needed to process PCD data.
Sub_Intv	MFPS	The beginning and ending major frame times corresponding to a predefined Sub_Intv time range.
WRS_Table_Info	MACS	The WRS scheme values.

Table 4.8 PCDS Interface Requirements - OUTPUT

Output Item Name	Destination	Description
Scene_Info	IDPS	The scene identification in accordance with the Worldwide Reference System (WRS) scheme
PCD_File	LDTS	A file containing the PCD Major Frames received during a subinterval, on a full PCD Cycle basis.
PCD_Acct	MACS	The statistics that are gathered from the processing of PCD_Info.
Drift_Time	IDPS	The equation that provides the difference between the spacecraft time and the actual time of the ETM+ major frame.
PCD_Setup_Status	MACS	The status of validating the PCD parameters and thresholds.
PCD_Assemble_Cycle_Status	MACS	Errors that are detected during the processing of PCD data.

4.7.1.3 Detailed Functional Requirements

This section contains the process specifications for the lowest level processes in the PCDS data flow diagrams.

NAME:
4.1;15

TITLE:
Validate PCD Parameters

INPUT/OUTPUT:
Valid_WRS_Parms : data_out
Valid_Scene_Parms : data_out
PCD_Setup_Status : data_out
Valid_PCD_Thres : data_out
Valid_PCD_Parms : data_out
PCD_Parms : data_in
PCD_Thresholds : data_in
Current_Time : data_in

BODY:
Description of Process

Validate PCD Parameters validates and stores user input parameters that are used to process PCD data.

Assumptions

Preconditions

None.

Postconditions

MACS is notified if there are invalid parameters or thresholds.

Constraints

The parameters must conform to predefined thresholds
The processing of data for a contact period will not be interrupted to process updates to the parameters and thresholds.

Functional Breakdown

Validate that PCD_Thresholds are within the following limits:

PCD_Directive.PCD_Thresholds.Att_Lower_Bounds
< PCD_Directive.PCD_Thresholds.Att_Upper_Bounds
PCD_Directive.PCD_Thresholds.Ephem_Position_Lower
< PCD_Directive.PCD_Thresholds.Ephem_Position_Upper
PCD_Directive.PCD_Thresholds.Ephem_Velocity_Lower
< PCD_Directive.PCD_Thresholds.Ephem_Velocity_Upper
PCD_Directive.PCD_Thresholds.Num_Failed_Votes >0
PCD_Directive.PCD_Thresholds.Num_Missing_Data_Words >0
Store the validated PCD_Thresholds entries in Valid_PCD_Thres.

Validate that PCD_Parms entries are within the following limits:

PCD_Directive.PCD_Parms.Frame_Fill_Values >=0
Store the validated PCD_Parms entries in Valid_PCD_Parms.

Validate that the scene setup PCD_Parms entries are within the following limits:

PCD_Directive.PCD_Parms.ETM_Plus_To_Body_Trans >=-1 and <=1
PCD_Directive.PCD_Parms.Time_Per_Orbit > 0
PCD_Directive.PCD_Parms.Mission_Start_Time >= Current_Time

PCD_Directive.PCD_Parms.ETM_Plus_LOS_x >=TBD
PCD_Directive.PCD_Parms.ETM_Plus_LOS_y >=TBD
PCD_Directive.PCD_Parms.ETM_Plus_LOS_z >=TBD
PCD_Directive.PCD_Parms.Semi_Major_Axis >=0
PCD_Directive.PCD_Parms.Semi_Minor_Axis >=0
Store the valid scene setup PCD_Parms entries in
Valid_Scene_Parms.

Validate that WRS PCD_Parms entries are within the following limits:
PCD_Directive.PCD_Parms.WRS_Row_Nominal 0 to 248
PCD_Directive.PCD_Parms.Latitude >=-90 and <=90
For each path corresponding to the row verify that:
PCD_Directive.PCD_Parms.WRS_Path_Nominal 0 to 233
PCD_Directive.PCD_Parms.Longitude >=-180 and <=180
Calculate Upper_Left_Corner_Latitude,
Calculate Upper_Left_Corner_Longitude,
Calculate Upper_Right_Corner_Latitude,
Calculate Upper_Right_Corner_Longitude,
Calculate Lower_Left_Corner_Latitude,
Calculate Lower_Left_Corner_Longitude,
Calculate Lower_Right_Corner_Latitude, and
Calculate Lower_Right_Corner_Longitud.
Store the validated PCD_Parms and calculated data entries in
Valid_WRS_Parms.

Send a message to the MACS specifying the names and values of any parameters or thresholds that are in error. (PCD_Status.PCD_Setup_Status)

Reusability
None.

NAME:
4.2.1;4

TITLE:
Extract Info Word

INPUT/OUTPUT:
PCD_Info : data_in
Partial_Rep_Info_Word : data_inout
Rep_Info_Word : data_out

BODY:
Description of Process
Extract Info Word extracts the triplicate PCD information word from the PCD_Info.PCD_Byte stream.

Assumptions
Preconditions
The first four PCD bytes have been extracted from the PCD/Status section of every VCDU and passed to the PCDS.
A count of missing VCDUs must be provided by the Major Frame Processing Subsystem (MFPS).
A token indicating the end of a contact period must be provided by the MFPS immediately after all PCD_Bytes for the contact period has been sent.
Post conditions
The bytes following the information word sync pattern will be stored as a Partial_Rep_Info_Word, if necessary, until three words have been obtained and assigned to Rep_Info_Word.
If VCDUs are missing, then the Rep_Info_Word.Info_Word_Missing value is set to reflect the number of missing information words.
Constraints
None.

Functional Breakdown
Search the PCD_Info.PCD_Bytes for an information word sync pattern (0x16) which identifies the beginning of a PCD Data Word Cycle.
Maintain a count of the number of words received for each PCD Data Word Cycle.
If an information word sync pattern is found, inspect the next three words for a fill word pattern (0x32) followed by another information word sync pattern (0x16).
If the fill-sync pattern is found within the three words then use any data word(s) that occur prior to the fill-sync pattern, assign null value(s) to the information words that were not available for the Rep_Info_Word and increment the Rep_Info_Word.Info_Word_Missing.
If the fill-sync pattern is not found within the three words then extract the three data words and store as Rep_Info_Word. The Rep_Info_Word.Info_Word_Missing is assigned a value that reflects the number of information words that were not available.
If PCD_Info.Num_Missing_VCDUs indicates that missing VCDUs were encountered and the number of words received per PCD Data Word Cycle exceeds

nine (9) then assign a Null value to the three information words of Rep_Info_Word and increment Rep_Info_Word.Info_Word_Missing.

If PCD_Info.End_Of_Contact_Flag indicates that the contact period has ended, then

Retrieve any data contained in Partial_Rep_Info_Word and assign them to Rep_Info_Word.

If the number of remaining Partial_Rep_Info_Word is < 3 then assign a null value to Rep_Info_Word to represent the missing information word(s).

Assign a value to Rep_Info_Word.Info_Word_Missing that reflects the number of missing information words.

If the number of remaining Partial_Rep_Info_Word is = 3 then assign the Partial_Rep_Info_Word to Rep_Info_Word.

Set Rep_Info_Word.Contact_Ended.

Place the PCD_Info.Sub_Intv_Id into the Rep_Info_Word.Sub_Intv_Id.

Output Rep_Info_Word.

Reusability

None.

NAME:
4.2.2;5

TITLE:
Determine Majority Info Word

INPUT/OUTPUT:
PCD_Info_Word : data_out
Rep_Info_Word : data_in

BODY:
Description of Process
Determine Majority Info Word derives the PCD_Info_Word from the Rep_Info_Word.

Assumptions
Preconditions
The value of Failed_PCD_Votes is equal to the number of failed attempts to obtain a majority information word for the current Sub_Intv.

Post conditions
None.

Constraints
None.

Functional Breakdown
If Rep_Info_Word.Contact_Ended is set then set the PCD_Info_Word.Flush_Minor_Frames.
If the Rep_Info_Word.Info_Word_Missing value indicates that each of the three information words are missing then set the PCD_Info_Word.Info_Word_Missing value to one and assign a null value to the PCD_Info_Word.Info_Word.
If the Rep_Info_Word.Info_Word_Missing value is >=1<3 then assign zero to PCD_Info_Word.Info_Word_Missing and assign (TBD) word from Rep_Info_Word to PCD_Info_Word.Info_Word.
If Rep_Info_Word.Info_Word_Missing indicates that there are no missing information words then perform a majority vote using Rep_Info_Word.Info_Word_1, Rep_Info_Word.Info_Word_2, and Rep_Info_Word.Info_Word_3. The majority word will be assigned to the PCD_Info_Word.Info_Word.
If all three values of Rep_Info_Word are different then set PCD_Info_Word.Majority_Vote_Failure. Assign (TBD) word of Rep_Info_Word to PCD_Info_Word.Info_Word.
Place the Rep_Info_Word.Sub_Intv_Id into the PCD_Info_Word.Sub_Intv_Id.
Output PCD_Info_Word.

Reusability
None.

NAME:
4.3.1

TITLE:
Assemble Minor Frames

INPUT/OUTPUT:
Failed_Votes : data_inout
Partial_PCD_Minor_Frame : data_inout
Sub_Intv_Id : data_out
PCD_Assemble_Cycle_Status : data_out
PCD_Minor_Frame : data_out
Minor_Frame_Acct : data_out
PCD_Info_Word : data_in
Valid_PCD_Thres : data_in
PCD_Frame_Fill : data_in

BODY:
Description of Process
Assemble Minor Frames builds the PCD_Minor_Frames using the PCD_Info_Word and calculates the Minor Frame related statistics.

Assumptions
Preconditions
A predefined value that represents missing VCDUs and failed attempts to perform a majority vote must exist.

Post conditions
None.

Constraints
None.

Functional Breakdown
If the PCD_Info_Word.Info_Word_Missing indicates that a valid information word is available then append PCD_Info_Word.Info_Word to the current Partial_PCD_Minor_Frame.
If the PCD_Info_Word.Info_Word_Missing indicates that a valid information word is not available then append a predefined fill pattern, Valid_PCD_Parms.PCD_Frame_Fill, to the current Partial_PCD_Minor_Frame.
If the PCD_Info_Word.Flush_Minor_Frame.Contact_Ended flag indicates that the contact period has ended, then
Use Valid_PCD_Parms.PCD_Frame_Fill to complete the current Partial_PCD_Minor_Frame.
Output the completed PCD_Minor_Frame followed by a PCD_Minor_Frame that has a null value, which indicates that the contact period has ended.
If the PCD_Info_Word is a PCD_Minor_Frame.Info_Word.Sync_Word, and the next two PCD_Info_Words are PCD_Minor_Frame.Info_Word.Sync_Word, then
Use Valid_PCD_Parms.PCD_Frame_Fill to complete the current Partial_PCD_Minor_Frame
Output the PCD_Minor_Frame.

Initialize the Partial_PCD_Minor_Frame
Append the three PCD_Minor_Frame.Info_Word.Sync_Word to the new Partial_PCD_Minor_Frame.

Calculate the number of missing information words by accumulating the PCD_Info_Word.Info_Word_Missing.
If the number of missing data words exceeds the Valid_PCD_Thresh.Num_Missing_Data_Words then generate PCD_Errors.Data_Missing_Message and send to MACS.

Calculate the number of PCD Minor Frames that have sync errors and store as Minor_Frame_Acct.Num_PCD_MNF_Sync_Errors.
Calculate the number of PCD Minor Frames that have been filled and store as Minor_Frame_Acct.Num_PCD_Filled_MNF.
Calculate the number of Failed_Votes accumulating the PCD_Info_Word. Majority_Vote_Failure flags and store as Minor_Frame_Acct. Failed_PCD_Votes.
Output the Minor_Frame_Acct
If the number of Failed_Votes exceeds the Valid_PCD_Thresh.Num_Failed_Votes then generate a PCD_Status. PCD_Assemble_Cycle_Status.Failed_Votes_Message and send to MACS.
Output the PCD_Info_Word.Sub_Intv_Id.

Reusability
None.

NAME:
4.3.2;7

TITLE:
Assemble Major Frames

INPUT/OUTPUT:
Partial_PCD_Major_Frame : data_inout
PCD_Major_Frame : data_out
Major_Frame_Acct : data_out
PCD_Minor_Frame : data_in
Valid_PCD_Parms : data_in

BODY:
Description of Process
Assemble Major Frames builds the PCD_Major_Frame using an accumulation of each PCD_Minor_Frame.

Assumptions
Preconditions
The Major Frame Id of the first PCD Minor Frame of every PCD Major Frame will be used as the key value for the entire PCD Major Frame. Future Major Frame Ids within the PCD Major Frame will be compared to the Major Frame Id of the first PCD Minor Frame.
If a predefined fill pattern occupies word 65 in the PCD Minor Frame then the PCD Minor Frame will be included in the building of the current PCD Major Frame.
A predefined pattern that indicates missing PCD Minor Frames must exist in data store.
Predefined quality indicators for ephemeris data points and attitude data points must exist in a data store.

Post conditions
None.

Constraints
None.

Functional Breakdown
If the value of the PCD_Minor_Frame is null then use Valid_PCD_Parms.PCD_Frame_Fill to complete the current Partial_PCD_Major_Frame.
Output the completed PCD_Major_Frame followed by a PCD_Major_Frame with a null value to indicate that the contact period has ended.
Compare the PCD_Minor_Frame.Info_Word.Major_Frame_Id in word 65 of the new PCD_Minor_Frame to the Partial_PCD_Major_Frame. PCD_Minor_Frame.Info_Word.Major_Frame_Id Word 65 to determine if the new PCD_Minor_Frame belongs to the current Partial_PCD_ Major_Frame.
If the new PCD_Minor_Frame belongs to the current Partial_PCD_Major_ Frame, then
Append the new PCD_Minor_Frame to the current Partial_PCD_

Major_Frame.
If the new PCD_Minor_Frame does not belong to the current Partial_PCD_Major_Frame, then
Use Valid_PCD_Parms.PCD_Frame_Fill to complete the current Partial_PCD_Major_Frame.
Output the completed PCD_Major_Frame.
Append the new PCD_Minor_Frame to a new Partial_PCD_Major_Frame.
Evaluate the ephemeris and attitude data points of each PCD_Major_Frame to determine the quality using
Valid_PCD_Parms.Ephem_Position_Upper,
Valid_PCD_Parms.Ephem_Position_Lower,
Valid_PCD_Parms.Ephem_Velocity_Upper,
Valid_PCD_Parms.Ephem_Velocity_Lower,
Valid_PCD_Parms.Att_Lower_Bounds,
Valid_PCD_Parms.Att_Upper_Bounds
as thresholds to determine the number of rejected ephemeris and attitude data points and store in Major_Frame_Acct.Num_Rejected_ADP and Major_Frame_Acct.Num_Rejected_EDP.
Calculate the number of PCD Major Frames that contain a Valid_PCD_Parms.PCD_Frame_Fill and store as Major_Frame_Acct.Num_PCD_Filled_MJF.
Calculate the number of missing ephemeris and attitude data points and store in Major_Frame_Acct.Num_Avail_ADP.
Calculate the number of available ephemeris and attitude data points and store in Major_Frame_Acct.Num_Avail_EDP.

Reusability
None.

NAME:
4.3.3;12

TITLE:
Build PCD Cycles

INPUT/OUTPUT:
Cycle_Acct : data_inout
Partial_PCD_Cycle : data_inout
Full_PCD_Cycle : data_out
PCD_Frame_Info : data_out
Minor_Frame_Acct : data_in
Major_Frame_Acct : data_in
PCD_Major_Frame : data_in
PCD_Frame_Fill : data_in
Sub_Intv_Id : data_in

BODY:
Description of Process
Build PCD Cycles uses PCD_Major_Frame to build a Full_PCD_Cycle.
The PCD_Frame_Info is collected and reported for each
Full_PCD_Cycle on a subinterval basis.

Assumptions
Preconditions
None.

Post conditions
Partial PCD Cycles will be filled using a PCD_Frame_Fill.(TBR).

Constraints
None.

Functional Breakdown
Place the Sub_Intv_Id into the Full_PCD_Cycle.Sub_Intv_Id.
Collect Minor_Frame_Acct which is received on a PCD Minor Frame basis and
store the information into the Cycle_Acct store.
Collect Major_Frame_Acct which is received on a PCD Major Frame basis and
store the information into the Cycle_Acct store.
Use the PCD_Major_Frame.PCD_Minor_Frame.Info_Word.Major_Frame_Id word 65
to determine the Major_Frame_Id.
If the Major_Frame_Id indicates that the PCD_Major_Frame is part of the
current Partial_PCD_Cycle then append the PCD_Major_Frame to the
current Partial_PCD_Cycle.
If the Major_Frame_Id indicates that the PCD_Major_Frame is not part of
the current Partial_PCD_Cycle then use PCD_Frame_Fill to
complete the current Partial_PCD_Cycle.
Place the Partial_PCD_Cycle into the Full_PCD_Cycle
Output the Full_PCD_Cycle.
If the Major_Frame_Id contains a value of zero then start a new
Partial_PCD_Cycle, appending the current PCD_Major_Frame.
If the Major_Frame_Id contains a value other than zero then
discard the current PCD_Major_Frame.
If the PCD_Major_Frame has a null value then the current Partial_PCD_Cycle

will be completed using PCD_Frame_Fill.
A Full_PCD_Cycle will be generated with a null value to indicate that the contact period has ended.
Output the Full_PCD_Cycle.

Format the Cycle_Acct and store as PCD_Frame_Info for the Full_PCD_Cycle and assign the Sub_Intv_Id to PCD_Frame_Info.Sub_Intv_Id.

Store the PCD_Frame_Info in PCD_Acct identified by Sub_Intv_Id.

Reusability

None.

NAME:
4.4.1;13

TITLE:
Compute Position MJF Time

INPUT/OUTPUT:
Cal_Info : data_inout
MJF_Time_And_Position : data_inout
Major_Frame_Time : data_in
Full_PCD_Cycle : data_in
MF_Start_Time : data_in
MF_Stop_Time : data_in

BODY:
Description of Process
Compute Position MJF Time computes the ETM+ Major Frame Time for each position within the Full_PCD_Cycle and determines the calibration door activity status for each position.

Assumptions
Preconditions
A predefined, fixed time length indicating the period of time between the generation of PCD Major Frames must exist.

Postconditions
Interpolated attitude and ephemeris data points will not be included in the PCD_File but will be used for latitude and longitude calculations.
The Cal_Info.Calibration_Door_Activity_Status is the same for each PCD_Major_Frame within a Full_PCD_Cycle.

If the spacecraft time of the Full_PCD_Cycle is greater than the corresponding Sub_Intv.MF_Stop_Time then the Full_PCD_Cycle will be held until either the sub-interval changes or additional MF_Start_Time and MF_Stop_Times are reported for the current sub-interval.

Constraints
None.

Functional Breakdown
If the Full_PCD_Cycle.Sub_Intv_Id is not the same as the MJF_Time_And_Position.Sub_Intv_Id then the MJF_Time_And_Position data store is initialized to reflect the different sub-interval.
Store the Full_PCD_Cycle.Sub_Intv_Id into the MJF_Time_And_Position.Sub_Intv_Id.
For each PCD_Major_Frame in the Full_PCD_Cycle do the following:

Extract the Ephemeris from the PCD_Major_Frame.
Extract the Attitude from each PCD_Major_Frame.
Extract the Cal_Door_Activity_Status from PCD_Major_Frame 3.
Use the second and third bits of word 72 to determine the status of the calibration door and store as Cal_Info.

Cal_Door_Activity_Status.
Extract the spacecraft time from PCD_Major_Frame 1.PCD_Minor_Frame 96-102.Word 72.
Interpolate any missing attitude or ephemeris points using data points and time from a previous PCD_Major_Frame within the Sub_Intv.
Use the Major_Frame_Time that is closest to the PCD spacecraft time as the time stamp for the first PCD_Major_Frame of the Full_PCD_Cycle.
Compute the Major_Frame_Time for the remaining PCD_Major_Frame of the Full_PCD_Cycle by incrementing the Major_Frame_Time by a predefined time length, insuring that the computed time does falls within the Sub_Intv.MF_Start_Time and the Sub_Intv. MF_Stop_Time.
Assign the attitude points to MJF_Time_And_Position.Attitude.
Assign the ephemeris data points to MJF_Time_And_Position.Ephemeris.
Assign the Major_Frame_Time to MJF_Time_And_Position. Major_Frame_Time.
Assign the Cal_Info.Cal_Door_Activity_Status to MJF_Time_And_Position. Cal_Door_Activity_Status.

Reusability
None.

NAME:
4.4.2;14

TITLE:
Compute Latitude And Longitude

INPUT/OUTPUT:
Lat_And_Long : data_out
MJF_Time_And_Position : data_in
Valid_Scene_Parms : data_in

BODY:
Description of Process
Compute Latitude and Longitude computes the latitude and longitude for each PCD Major Frame time and position.

Assumptions
Preconditions
The transformation parameters must exist in the Valid_Scene_Parms.

Postconditions
None.

Constraints:
None.

Functional Breakdown
Place the MJF_Time_And_Position.Sub_Intv_Id into Lat_And_Long.Sub_Intv_Id.
For each MJF_Time_And_Position associated with the MJF_Time_And_Position.Sub_Intv_Id,
Compute the Lat_And_Long.Latitude using the MJF_Time_And_Position.Time,MJF_Time_And_Position.Position, Valid_Scene_Parms.ETM_Plus_To_Body_Trans, Valid_Scene_Parms.Semi_Major_Axis, Valid_Scene_Parms.Semi_Minor_Axis and Valid_Scene_Parms.ETM_Plus_LOS as input to the Latitude and Longitude Algorithm located in Appendix C.
Compute the Lat_And_Long.Longitude using the MJF_Time_And_Position.Time, MJF_Time_And_Position.Position, Valid_Scene_Parms.ETM_Plus_To_Body_Trans, Valid_Scene_Parms.Semi_Major_Axis, Valid_Scene_Parms.Semi_Minor_Axis, Valid_Scene_Parms.ETM_Plus_LOS_x, Valid_Scene_Parms.ETM_Plus_LOS_y, and ETM_Plus_LOS_z, as input to the Latitude and Longitude Algorithm located in Appendix C.
Assign the MJF_Time_And_Position.Cal_Door_Activity_Status to the Lat_And_Long.Cal_Door_Activity_Status.
Output Lat_And_Long.

Reusability
The algorithm for the subroutine, JGHAX, will be re-used. JGHAX is used to compute the Greenwich Hour Angle (GHA) which is needed to compute the longitude.

NAME:
4.4.3

TITLE:
Determine WRS Scene Coordinates

INPUT/OUTPUT:
Previous_Lat_And_Long : data_inout
PCD_Scene_Count : data_inout
Nominal_Scene_Coords : data_out
Actual_Center_Coords : data_out
Valid_WRS_Parms : data_in
Lat_And_Long : data_in
Major_Frame_Time : data_in
MF_Start_Time : data_in
MF_Stop_Time : data_in

BODY:
Description of Process

Determine Scene Coordinates uses the actual latitudes and longitudes to retrieve the closest related latitude, longitude, WRS Row, and WRS Path from the WRS Table.

Assumptions

Preconditions

None.

Postconditions

If the current Lat_And_Long.Major_Frame_Time is greater than the corresponding Sub_Intv.MF_Stop_Time then the current Lat_And_Long will be held until either the sub-interval changes or additional MF_Start_Time and MF_Stop_Times are reported for the current subinterval.

Constraints

None.

Functional Breakdown

If the Lat_And_Long.Sub_Intv_Id has changed from the Previous_Lat_And_Long.Sub_Intv_Id then initialize the PCD_Scene_Count and the Previous_Lat_And_Long data store.

Use the Lat_And_Long.Latitude and the Lat_And_Long.Longitude to locate the closest related scene center lat and long in the Valid_WRS_Parms.

If a Lat_And_Long.Latitude and Lat_And_Long.Longitude indicate that a WRS scene center has not been crossed then store the values as Previous_Lat_And_Long.Latitude and Previous_Lat_And_Long.Longitude.

If a Lat_And_Long.Latitude and Lat_And_Long.Longitude indicate that a WRS scene center has been crossed then retrieve the Previous_Lat_And_Long.Lat_And_Long.Latitude and Previous_Lat_And_Long.Lat_And_Long.Longitude to confirm that a WRS scene center has been crossed.

Maintain a count of the number of scenes identified for each Lat_And_Long.Sub_Intv_Id as PCD_Scene_Count.

Use the Lat_And_Long values and the Previous_Lat_And_Long values to calculate the Actual_Center_Coords.Latitude and the Actual_Center_Coords.Longitude by interpolation.

Store the Path and the Row of the scene center that has been crossed as the Nominal_Scene_Coords.WRS_Path_Nominal, and the Nominal_Scene_Coords.WRS_Row_Nominal.

Correlate the Lat_And_Long.Major_Frame_Time to the closest Major_Frame_Time within the Sub_Intv.MF_Start_Time and Sub_Intv.MF_Stop_Time identified by the Lat_And_Long.Sub_Intv_Id.

Determine the sequence of the closest Major_Frame within the Sub_Intv and store the value as Nominal_Scene_Coords.Scene_Center_Scan_Num.

Store the Lat_And_Long.Sub_Intv_Id as Actual_Center_Coords.Sub_Intv_Id.

Store the Lat_And_Long.Major_Frame_Time as Actual_Center_Coords.Scene_Center_Time.

Store the Lat_And_Long.Major_Frame_Time as Nominal_Scene_Coords.Scene_Center_Time.

Store the PCD_Scene_Count as Nominal_Scene_Coords.PCD_Scene_Count

Use PCD_Scene_Count to store a unique scene number to Nominal_Scene_Coords.Sub_Intv_Scene_Num.

Store the Lat_And_Long.Sub_Intv_Id as Nominal_Scene_Coords.Sub_Intv_Id.

Store the Lat_And_Long.Cal_Door_Activity_Status as Nominal_Scene_Coords.Cal_Door_Activity_Status.

Reusability

None.

NAME:
4.4.4;5

TITLE:
Compute Horizontal Display Shift

INPUT/OUTPUT:
Horizontal_Display_Shift : data_out
Nominal_Scene_Coords : data_in
Actual_Center_Coords : data_in

BODY:
Description of Process
Compute Horizontal Shift Display computes the difference between the actual scene center location and the nominal scene center location.

Assumptions
Preconditions
None.

Postconditions
None.

Constraints
None.

Functional Breakdown
Calculate the difference in distance between the Nominal Scene Center position and the Actual Scene Center position in meters.
Output the calculated Horizontal_Display_Shift.

Reusability
None.

NAME:
4.4.5;4

TITLE:
Calculate Sun Position

INPUT/OUTPUT:
Actual_Center_Coords : data_in
Sun_Elevation : data_out
Sun_Azimuth : data_out

BODY:
Description of Process
Calculate Sun Position calculates the sun elevation and the sun azimuth for each WRS scene.

Assumptions
Preconditions
None.

Postconditions
None.

Constraints
None.

Functional Breakdown
Use the time from the Actual_Center_Coords.Scene_Center_Time to calculate the sun vector.
Compute the Sun_Elevation using Actual_Center_Coords.Scene_Center_Time, Actual_Center_Coords.Latitude, Actual_Center_Coords.Longitude, and the sun vector as input to the Sun Elevation Algorithm located in Appendix C.
Compute the Sun_Azimuth using Actual_Center_Coords.Scene_Center_Time, Actual_Center_Coords.Latitude, Actual_Center_Coords.Longitude, and the sun vector as input to the Sun Azimuth Algorithm located in Appendix C.

Reusability
The algorithm for the subroutine, JGHAX, will be re-used.
JGHAX is used to compute the Greenwich Hour Angle (GHA) which is needed to compute the Sun_Azimuth and the Sun_Elevation.

NAME:
4.4.6;13

TITLE:
Report Scene Info

INPUT/OUTPUT:
Scene_Parms : data_out
Scene_Info : data_out
Horizontal_Display_Shift : data_in
Nominal_Scene_Coords : data_in
Sun_Elevation : data_in
Sun_Azimuth : data_in

BODY:
Description of Process
Report Scene Info formats and reports the Scene Id for use by the Image Data Processing Subsystem (IDPS) and the Scene Parms to be included in the PCD Accounting file.

Assumptions
Preconditions
None.

Postconditions
None.

Constraints
None.

Functional Breakdown
Extract the Nominal_Scene_Coords.Sub_Intv_Id to obtain the associated subinterval.
Extract the Nominal_Scene_Coords.PCD_Scene_Count to determine the number of scenes identified.
Format Nominal_Scene_Coords.Sub_Intv_Id,
Nominal_Scene_Coords.PCD_Scene_Count,
Nominal_Scene_Coords.WRS_Path_Nominal,
Nominal_Scene_Coords.WRS_Row_Nominal,
Nominal_Scene_Coords.Scene_Center_Time,
Nominal_Scene_Coords.Scene_Center_Scan_Num,
Nominal_Scene_Coords.Sub_Intv_Scene_Num,
Nominal_Scene_Center.Cal_Door_Activity_Status,
Sun_Azimuth,
Sun_Elevation, and
Horizontal_Display_Shift
as Scene_Parms.
Store Scene_Parms into PCD_Acct identified by Nominal_Scene_Coords.Sub_Intv_Id
Store Nominal_Scene_Coords.PCD_Scene_Count as Scene_Info.PCD_Scene_Count.
Format Nominal_Scene_Coords.WRS_Path_Nominal,
Nominal_Scene_Coords.WRS_Row_Nominal,
Sun_Elevation, and

Nominal_Scene_Coords.Scene_Center_Time
as Scene_Info.Scene_Id.
Output the Scene_Info.

Reusability
None.

NAME:
4.5;5

TITLE:
Extract Major Frame Info

INPUT/OUTPUT:
Drift_Time : data_out
Sub_Intv_Id : data_in
Full_PCD_Cycle : data_in

BODY:
Description of Process
Extract Major Frame Info extracts the drift rate from each Full_PCD_Cycle.

Assumptions
Preconditions
None.

Post conditions
None.

Constraints
Each Full_PCD_Cycle must contain the complete Drift_Rate.

Functional Breakdown
Extract the Drift_Time from the first PCD Major Frame of each Full_PCD_Cycle.
Assign the current Full_PCD_Cycle.Sub_Intv_Id to Drift_Time. Sub_Intv_Id.
Assign the drift rate to Drift_Time.Drift_Rate.
Output the Drift_Time.

Reusability
None.

NAME:
4.6;14

TITLE:
Create PCD File

INPUT/OUTPUT:
Current_Orbit : data_inout
PCD_File_Info : data_out
Qualified_PCD_Cycle : data_out
MF_Start_Time : data_in
Major_Frame_Time : data_in
Time_Per_Orbit : data_in
Full_PCD_Cycle : data_in
MF_Stop_Time : data_in
LPS_Configuration : data_in

BODY:
Description of Process
Create PCD File creates the PCD_File using Full_PCD_Cycle and calculates the PCD_File related statistics on a Sub_Intv basis.

Assumptions
Preconditions
The Valid_Scene_Parms.Mission_Start_Time is stored as Current_Orbit.Orbit_Time at the beginning of the LPS mission.
Zero is stored as the Current_Orbit.Orbit_Num at the beginning of the LPS mission.
The PCD_File_Name consists of the LPS_Configuration. File_Version_Number, Sub_Intv.MF_Start_Time, and the PCD_File_Type.

Post conditions
None.

Constraints
None.

Functional Breakdown
If a PCD_File is open then the Full_PCD_Cycle is written to the current PCD_File.
If a PCD_File is not open then a new PCD_File is opened, assigned a PCD_File_Name, and the current Full_PCD_Cycle will be written to the new PCD_File. The PCD_File_Name of the new PCD_File will be stored.
If the Full_PCD_Cycle has a null value which indicates that a contact period has ended, then
Format and store PCD_File_Info.PCD_File_Name, PCD_File_Info.Num_PCD_MJF, and PCD_File_Info.First_PCD_MJF_Time
Close the current PCD_File.
Extract the spacecraft time from Full_PCD_Cycle.PCD_Major_Frame 0.
If the Full_PCD_Cycle is the first Full_PCD_Cycle for the current Full_PCD_Cycle.Sub_Intv_Id then store the spacecraft time as

the PCD_File_Info.First_PCD_MJF_Time.
Compare the PCD_File_Info.First_PCD_MJF_Time to the Current_Orbit.Orbit_Time to determine the current orbit.
If the PCD_File_Info.First_PCD_MJF_Time is <= the Current_Orbit.Orbit_Time then store the Current_Orbit.Orbit_Num as PCD_File_Info.Orbit_Num.
If the PCD_File_Info.First_PCD_MJF_Time is > the Current_Orbit.Orbit_Time then increment the Current_Orbit.Orbit_Num by one, increment the Current_Orbit.Orbit_Time by the Valid_Scene_Parms.Time_Per_Orbit and store the Current_Orbit.Orbit_Num as PCD_File_Info.Orbit_Num.

Compare the spacecraft time to the current Sub_Intv.MF_Start_Time and Sub_Intv.MF_Stop_Time time range, identified by Full_PCD_Cycle.Sub_Intv_Id, to determine if the Full_PCD_Cycle is a Qualified_PCD_Cycle for the current Sub_Intv.

If the Full_PCD_Cycle is a Qualified_PCD_Cycle for the current Sub_Intv, then
Store the Qualified_PCD_Cycle in PCD_File.
Calculate the PCD_File_Info.Num_PCD_MJF.

If the Full_PCD_Cycle is not a Qualified_PCD_Cycle for the current Sub_Intv then format and store PCD_File_Info.PCD_File_Name, PCD_File_Info.Num_PCD_MJF, PCD_File_Info.First_PCD_MJF_Time, and PCD_File_Info.Orbit_Num as PCD_File_Info.
Close the current PCD_File.

Place the contents of PCD_File_Info into the PCD_Acct store identified by Full_PCD_Cycle.Sub_Intv_Id.

Reusability

None.

4.7.2 Performance Requirements

The following list summarizes the performance requirements allocated to the PCDS:

4.7.2.1 The PCDS software on each LPS string shall provide the capability to process the equivalent of any combination of the format 1 and format 2 portions of 125 Landsat 7 ETM+ scenes of wideband data per day (approximately 25-30 GB per day).

4.7.2.2 The PCDS software on each LPS string shall process unpacked PCD data at a minimum rate of not less than 3.2 kilobits per second (Kbps) (based on a minimum raw wideband throughput of 7.5 Mbps)

4.7.2.3 The PCDS software on each LPS string shall provide the capability to execute concurrently with all other LPS subsystems.

4.7.2.3.1 The PCDS software shall begin to process received raw wideband data immediately upon receipt of required inputs.

4.7.2.3.2 The PCDS software shall output a scene center identification, a sun azimuth at scene center value, and a sun elevation at scene center value within 240 seconds of the time of receiving all required inputs.

4.7.2.4 The PCDS software on each LPS string shall provide the capability to reprocess a maximum of 10% of a string's daily input volume of wideband data (approximately 12.5 scenes or 2.5-3 GB per day).

4.7.2.5 The PCDS software on each LPS string shall provide the capability to process unpacked PCD data at a daily average aggregate rate of 12.7 Kbps per second (Includes 10% of overhead due to reprocessing).

4.7.2.6 The PCDS software on each LPS string shall maintain data processing throughput performance for all Landsat 7 raw wideband data received with a BER of one bit error in 10^5 bits, without loss of level zero processed data and without retransmission. [

4.7.2.7 The PCDS software on each LPS string shall output any periodic processing status information that it generates with a maximum latency of 30 seconds between outputs.

4.8 Image Data Processing Subsystem (IDPS)

This subsystem is responsible for producing band and browse image files, generating cloud coverage assessment scores, and validating certain user-defined parameters that are necessary for the subsystem.

4.8.1 Functional Requirements

The following list summarizes the functional requirements allocated to the IDPS:

- validate and store the processing parameters received from MACS.
- generate monochrome or multiband browse data for each ETM+ image on a subinterval basis.
- generate band file for each band received on a subinterval basis.
- perform Automatic Cloud Cover Assessment (ACCA) for WRS scenes using predefined comparison values on scene quadrant and full scene basis.

4.8.1.1 Major Functions

The IDPS subsystem, in its Band File Generation process, takes aligned band data from MFPS, separates it by subinterval and band, and produces band files. In the Browse File Generation process, the aligned band data is separated by subinterval and then reduced by subsampling and wavelet algorithms to produce monochrome and multiband browse files. In the ACCA process, the aligned band data is separated by scenes, and then an ACCA algorithm is used on the data to produce ACCA scores. Certain parameters used in the above processes are also validated before their use in the Validate Band Parameters process. Finally, accounting information is produced for metadata generation and the band and browse files are made available to the LDTS.

The major functions of IDPS are depicted in the following data flow diagrams.

Figure 4-15
IDPS - DFD 5.0

Figure 4-16
IDPS - DFD 5.2

Figure 4-17
IDPS - DFD 5.4

4.8.1.2 Interface Requirements

The following two tables summarize the interface requirements for the IDPS:

Table 4.9 IDPS Interface Requirements - INPUT

Input Item Name	Source	Description
Aligned_Bands	MFPS	Bands aligned along band and detector
Status_Info	MFPS	Status information from the VCDU
Sub_Intv	MFPS	Subinterval id, start time and stop time
Drift_Time	PCDS	Difference between spacecraft time and actual time
Scene_Info	PCDS	Parameters used to calculate the scene identification
File_Version_Number	MACS	File identifier for reprocessing
IDP_Band_Parms	MACS	Operator-defined parameters specifying which bands to process, whether to use band 6 for format 1 or 2, reduction ratios for Browse and ACCA, and the ACCA method used

Table 4.10 IDPS Interface Requirements - OUTPUT

Output Item Name	Destination	Description
Browse_File	LDTS	Monochrome and/or Multiband reduced image files.
Band_File	LDTS	3 or 6 separate band files for bands 6, 7 & Pan or bands 1-6
IDP_Acct	MACS	Accounting information from Browse, Band and ACCA
IDP_Setup_Status	MACS	Status returned from validating IDP_Band_Parms
IDP_ACCA_Status	MACS	Status returned from ACCA

4.8.1.3 Detailed Functional Requirements

This section contains the process specifications for the lowest level processes in the IDPS data flow diagrams.

NAME:
5.1;10

TITLE:
Validate IDP Parameters

INPUT/OUTPUT:
IDP_Setup_Status : data_out
Valid_Band_Parms : data_out
IDP_Band_Parms : data_in

BODY:
Description of Process

This function validates one or more of the following values of IDP_Band_Parms: the four band parameters that indicate which bands to process for Browse (IDP_Band_Parms.Mono, IDP_Band_Parms.Multi1, IDP_Band_Parms.Multi2, IDP_Band_Parms.Multi3), and reduction ratios for performing the subsampling, wavelet and ACCA algorithms (IDP_Band_Parms.Subs, IDP_Band_Parms.Wave and IDP_Band_Parms.CCA_Ratio). If valid, these values are placed in the datastore Valid_Band_Parms along with the identifier of the ACCA algorithm used (IDP_Band_Parms.CCA_Method) which does not need to be validated.

Assumptions

Preconditions

One or more of the following values of IDP_Band_Parms have been entered by an operator and passed to the task: Four band parameters (IDP_Band_Parms.Mono, IDP_Band_Parms.Multi1, IDP_Band_Parms.Multi2, IDP_Band_Parms.Multi3),both a subsampling and wavelet reduction ratio (IDP_Band_Parms.Subs and IDP_Band_Parms.Wave), and two ACCA parameters (IDP_Band_Parms.CCA_Method and IDP_Band_Parms.CCA_Ratio). There are valid default values for each of the above parameters in the datastore Valid_Band_Parms.

Postconditions

The following valid values exist in the datastore Valid_Band_Parms: Four band parameters (Valid_Band_Parms. Mono, Valid_Band_Parms.Multi1, Valid_Band_Parms.Multi2, Valid_Band_Parms.Multi3),both a subsampling and wavelet reduction ratio (Valid_Band_Parms.Subs and Valid_Band_Parms.Wave) and two ACCA parameters (Valid_Band_Parms.CCA_Method and Valid_Band_Parms. CCA_Ratio).

Constraints

Any changes to the band parameters will take effect at the start of the next contact period, which is identified in Sub_Intv.Contact_Id.

Functional Breakdown

Get any values for IDP_Band_Parms that are passed to this subsystem and validate their values. Following are the valid values for each:

Mono 1-7 or Pan

Multi1	1-7 or Pan
Multi2	1-7 or Pan
Multi3	1-7 or Pan
Subs	2-16
Wave	2-16.
CCA_Ratio	4,8,16,32,48
CCA_Method	arbitrary string - no validation needed

Send status (IDP_Status.IDP_Setup_Status) back to MACS indicating the success or failure of the validations so that appropriate messages may be logged and displayed.

The following assignments are made only if a new value is passed to the subsystem and if that value is valid:

Valid_Band_Parms.Mono = IDP_Band_Parms.Mono
Valid_Band_Parms.Multi1 = IDP_Band_Parms.Multi1
Valid_Band_Parms.Multi2 = IDP_Band_Parms.Multi2
Valid_Band_Parms.Multi3 = IDP_Band_Parms.Multi3
Valid_Band_Parms.Subs = IDP_Band_Parms.Subs
Valid_Band_Parms_Wave = IDP_Band_Parms.Wave
Valid_Band_Parms.CCA_Method = IDP_Band_Parms.CCA_Method
Valid_Band_Parms.CCA_Ratio = IDP_Band_Parms.CCA_Ratio

Reusability

None.

NAME:
5.2.1;14

TITLE:
Reduce Image by Subsamples

INPUT/OUTPUT:
Browse_Store : data_inout
Overlay_Subs : data_out
Mono_Band_Subs : data_out
Multi_Band_Subs : data_out
Valid_Band_Parms : data_in
Aligned_Bands : data_in
Sub_Intv : data_in
Status_Info : data_in
Scene_Info : data_in

BODY:
Description of Process

This function reduces an image by saving only the pixels in every Nth row and Nth column. The process is called subsampling by N where N is a user-defined parameter.

Assumptions

Preconditions

There are four valid band parameters indicating which bands to process (Valid_Band_Parms.Mono, Valid_Band_Parms.Multi1, Valid_Band_Parms.Multi2, Valid_Band_Parms.Multi3),and subsampling and wavelet reduction ratios (Valid_Band_Parms.Subs and Valid_Band_Parms.Wave) stored in Valid_Band_Parms. Valid band parameters are 1-6. Aligned band data (Aligned_ Bands) has been passed to the subsystem and subinterval definitions are available in the datastore Sub_Intv. The Status_Info.Fmt_Id, which identifies the scene data as either 1 or 2, has been passed to the subsystem.

Post conditions

Multi_Band_Subs and/or Mono_Band_Subs have been created by subsampling. Mono_Band_Subs contains monochrome browse data from one predetermined band. Multi_Band_Subs, if it exists, contains multiband browse data from three predetermined bands. Multi_Band_Subs will only be created if Status_Info.Fmt_Id is 1.

Constraints

None.

Functional Breakdown

Get subinterval start and stop times (Sub_Intv.MF_Start_Time and Sub_Intv.MF_Stop_Time), and Sub_Intv.Contact_Id from the Sub_Intv datastore identified by Aligned_Bands. Sub_Intv_Id.

Get Valid_Band_Parms.Subs from datastore Valid_Band_Parms.

For each new contact period (when Sub_Intv.Contact_Id changes),
get the following parameters from datastore Val_Band_Parms:

Valid_Band_Parms.Mono
Valid_Band_Parms.Multi1
Valid_Band_Parms.Multi2
Valid_Band_Parms.Multi3

Place Aligned_Bands data into datastore Browse_Store until the
data for one subinterval has accumulated (when
Aligned_Bands.Sub_Intv_Id changes).
Then get from Browse_Store the Aligned_Bands data for one subinterval.

If Status_Info.Format_Id is 1, then
Save only the Aligned_Bands data where Aligned_Bands.Band_Num
matches one of the four band parameters in Valid_Band_Parms.
If one of the four band parameters in Valid_Band_Parms is 6
and if Valid_Band_Parms.Fmt is 1, then
Retain Band 6 for format 1 scene data only.

Subsample the Aligned_Bands data by saving the pixel from every Nth
row and Nth column where N is Valid_Band_Parms.Subs.

Do this where Aligned_Bands.Band_Num matches Valid_Band_Parms.Mono
to produce reduced monochrome image data contained in Mono_Band_Subs.

If Status_Info.Format_Id is 1, do this where Aligned_Bands.Band_Num
matches Valid_Band_Parms.Multi1, Valid_Band_Parms.Multi2, and
Valid_Band_Parms.Multi3 to produce reduced multiband
image data contained in Multi_Band_Subs.

For each subsampled pixel which represents data that includes a
scene center (identified by Scene_Info.Scene_Id), place the
Scene_Info.Scene_Id and the pixel location in Overlay_Subs.
Overlay_Marks.

Append to Overlay_Subs, Mono_Band_Subs, and Multi_Band_Subs the
Aligned_Bands.Sub_Intv_Id, the Sub_Intv.MF_Start_Time and
Sub_Intv.MF_Stop_Time identified by Aligned_Bands.Sub_Intv_Id,
and the browse image reduction ratios
(Valid_Band_Parms.Subs and Valid_Band_Parms.Wave).

Output the Mono_Band_Subs, Multi_Band_Subs, and Overlay_Subs.

Reusability

The prototype written by Lizz Pena and described in a paper
written in Summer, 1994, ("Landsat Browse Generation Using
Wavelets for Image Reduction") may be reused.

The algorithm to separate aligned band data by subinterval
may be reused for Band File Generation and ACCA.

NAME:
5.2.2;14

TITLE:
Reduce Image by Wavelets

INPUT/OUTPUT:
Browse_File : data_out
Browse_Acct : data_out
Multi_Band_Subs : data_in
Mono_Band_Subs : data_in
Wave : data_in
Overlay_Subs : data_in
File_Version_Number : data_in
MF_Start_Time : data_in

BODY:
Description of Process

This function reduces an image by replacing an N X N grid of pixels with one pixel where the intensity of that one pixel is the average intensity of the entire grid of pixels. N is a user-defined parameter.

Assumptions

Preconditions

Multi_Band_Subs and/or Mono_Band_Subs have been produced by subsampling and have been passed to the task.

There is a wavelet reduction ratio, Valid_Band_Parms.Wave, in the Valid_Band_Parms datastore.

Post conditions

Multi_Band_Wave and/or Mono_Band_Wave have been created using the wavelet algorithm.

Mono_Band_Wave contains monochrome browse data from one predetermined band.

Multi_Band_Wave, if it exists, contains multiband browse data from three predetermined bands.

Browse_Acct has been passed to the IDP_Acct datastore.

Constraints

If Multi_Band_Subs has not been passed to this task, Multi_Band_Wave will not be generated.

Browse file will not be created for format 2 data.

Functional Breakdown

Get wavelet reduction ratio Valid_Band_Parms.Wave from Valid_Band_Parms datastore.

Reduce Mono_Band_Subs by replacing each N X N grid of pixels with one pixel where the intensity of that one pixel is the average intensity of the entire grid of pixels.
N is Valid_Band_Parms.Wave.
The result is Mono_Band_Wave.

For each wavelet reduced pixel which contains a scene center

(identified in Overlay_Subs.Overlay_Marks),
Copy the Overlay_Subs.Overlay_Marks.Scene_Id to
Overlay_Wave.Overlay_Marks.Scene_Id and insert
the pixel location into Overlay_Wave.Overlay_Mark.
Pixel_Index.
Place Overlay_Wave and Mono_Band_Wave into Browse_File.

If Multi_Band_Subs has been passed to this task, then
Replace each N X N grid of pixels with one pixel where the
intensity of that one pixel is the average intensity of the
entire grid of pixels.
N is Valid_Band_Parms.Wave.
The result is Multi_Band_Wave.
Place Multi_Band_Wave into Browse_File.

Create the Mono_Browse_File_Name using LPS_Configuration.
File_Version_Number and Sub_Intv.MF_Start_Time identified
by the Mono_Band_Subs.Sub_Intv_Id.
If Multi_Band_Wave exists,
Create the Multi_Browse_File_Name using LPS_Configuration.
File_Version_Number and Sub_Intv.MF_Start_Time identified
by the Mono_Band_Subs.Sub_Intv_Id.

Place the Mono_Band_Subs.Sub_Intv_Id into Browse_Acct.Sub_Intv_Id.
Place Multi_Browse_File_Name and/or Mono_Browse_File_Name in
Browse_Acct.Browse_File_Names.
Place Browse_Acct into IDP_Acct identified by Browse_Acct.Sub_Intv_Id.

Reusability

The prototype written by Lizz Pena during the Summer, 1994,
("Landsat Browse Generation Using Wavelets for Image Reduction")
may be reused.

NAME:
5.3;10

TITLE:
Generate Band File

INPUT/OUTPUT:
Band_Store : data_out
Band_File : data_out
Band_Acct : data_out
Aligned_Bands : data_in
Drift_Time : data_in
Status_Info : data_in
Band_Store : data_in
MF_Start_Time : data_in
File_Version_Number : data_in

BODY:
Description of Process
This function generates one file for each band and appends the Drift_Time and Status_Info to each file. These files are produced on a subinterval basis.

Assumptions
Preconditions
None.

Postconditions
Three or six band files have been generated and the Drift_Time and Status_Info have been appended to each band file.
Band_Acct has been passed to the IDP_Acct datastore.

Constraints
None.

Functional Breakdown
Place Aligned_Bands data into datastore Band_Store until the data for one subinterval has accumulated (when Aligned_Bands.Sub_Intv_Id changes).
Get from Band_Store the Aligned_Bands for the current subinterval.

If Status_Info.Format_Id is 1, then
Place the Aligned_Bands data for bands 1-6 in six separate files, one file for each band.
If Status_Info.Format_Id is 2, then
Place the Aligned_Bands data for bands 6, 7, and Pan in three separate files, one for each band.

Append to each band file the Status_Info and Drift_Time.
The resulting files are App_Data_Band1, App_Data_Band2, ..., App_Data_Band7, App_Data_Pan.
Place these band files into Band_File.

Place Aligned_Bands.Sub_Intv_Id into Band_Acct.Sub_Intv_Id
If Status_Info.Format_Id is 1,
    Place band numbers 1-6 into Band_Acct.Bands_Present.
    Create the Band_File_Names for bands 1-6 using
    LPS_Configuration.File_Version_Number and Sub_Intv.
    MF_Start_Time identified by Aligned_Bands.Sub_Intv_Id.
If Status_Info.Format_Id is 2,
    Place band numbers 6, 7 and Pan in Bands_Present.
    Create the Band_File_Names for bands 6, 7 and Pan using
    LPS_Configuration.File_Version_Number and Sub_Intv.
    MF_Start_Time identified by Aligned_Bands.Sub_Intv_Id.

.

Place the Band_File_Names into Band_Acct.Band_File_Names.
Place Status_Info.Band_Gains into Band_Acct.Band_Gains.
Place Status_Info.Gain_Change_Flag into Band_Acct.Gain_Change_Flag.
Place the Band_Acct into the IDP_Acct store identified by Band_Acct.
    Sub_Intv_Id.

Reusability

The algorithm to separate aligned band data by subinterval may be reused for Browse File Generation and ACCA.

NAME:
5.4.1;9

TITLE:
Collect Scene Data

INPUT/OUTPUT:
PCD_Scene_Count : data_out
Aligned_Bands : data_out
IDP_ACCA_Status : data_out
Scene_Metadata : data_out
Scene_Data : data_out
Status_Info : data_in
Aligned_Bands : data_in
Scene_Info : data_in
Scene_Data : data_in

BODY:
Description of Process
This function collects the scene data (identified by the Scene_Info.Scene_Id) from the Aligned_Bands stream. Cloud Cover Assessment only works with format 1 data.

Assumptions
Preconditions
Scene_Id will be a valid identifier of a scene that is contained in the current Aligned_Bands.

Postconditions
Scene_Data will contain the entire data for the identified scene.
The Scene_Metadata will contain the data to uniquely identify the data.

Constraints
The Aligned_Bands data stream must contain the correct data that may be found using the Scene_Info. Scene_Id.
Cloud Cover Assessment is only performed on Format 1 data.
A full scene's worth of data must be available to be collected.

Functional Breakdown
Place Scene_Info.PCD_Scene_Count into the IDP_Acct.
PCD_Scene_Count identified by Aligned_Bands.Sub_Intv_Id.

For each identified scene (PCD_Scene_Count) in Scene_Info do the following:

If Status_Info.Format_Id is 1, i.e. bands 1-6 exist, do
Use Scene_Info.Scene_Id to find the location of the start of the scene.
Scene_Info.Scene_Id identifies the scene center.

Use an appropriate delta to find the scene start time.

Read Aligned_Bands until scene start time is found.

Place Aligned_Bands into Scene_Data_Store until the end
of the scene.
If the full scene is not in the subinterval, then
Create a IDP_ACCA_Status message indicating that
no CCA can be performed because there is not a
full scene's worth of data.
Otherwise,
Pull the Scene_Data from the Scene_Data_Store for
the scene and output.

Place Aligned_Bands.Sub_Intv_Id and Scene_Info.Scene_Id
into Scene_Metadata.
Output Scene_Metadata and Scene_Data.
Otherwise,
Create IDP_ACCA_Status indicating there the data was
of the incorrect format.

Reusability
Possible reuse of scene collection algorithm.

Generating and sending status messages. Every major function (and probably most minor functions) will need to send messages to the MACS.

NAME:
5.4.2;10

TITLE:
Generate Cloud Cover Assessment

INPUT/OUTPUT:
ACCA_Acct : data_out
IDP_ACCA_Status : data_out
Scene_Data : data_in
CCA_Method : data_in
Scene_Metadata : data_in
CCA_Ratio : data_in

BODY:
Description of Process

The Generate Cloud Cover Assessment function determines percentage of cloud coverage on a scene quadrant and full scene basis.

The function subsamples, using the Valid_Band_Parms.CCA_Ratio, and the scene data from specified bands (TBD).

Assumptions

Preconditions

The Scene_Data must contain all of the data for a scene.

Postconditions

Five percentage scores have been generated and put in ACCA:

one for each quadrant and an aggregate score for the scene.

Status messages are sent indicating that CCA has been performed on a specific scene.

Constraints

Cloud cover assessment is only given for full scenes. Fill within major frames is allowable; it just won't be used in calculating an assessment.

Functional Breakdown

Find the start of each quadrant in Scene_Data.

Use Scene_Metadata.Scene_Id.Sun_Elevation to calculate a threshold used by the assessment algorithm.

For each quadrant in the scene

Pull data values from the Scene_Data at the sampling ratio defined by Valid_Band_Parms.CCA_Ratio.

Examine data values against the cloud cover assessment thresholds or algorithm.

Compile the cloud cover percentage quadrant score and place in ACCA_Acct.ACCA.CC_Quadrant#_Score.

Average the quadrant assessments for a scene score and place into
ACCA_Acct.ACCA.CCA_Aggregate_Score

Place Valid_Band_Parms.CCA_Method into ACCA_Acct.
Place Scene_Metadata.Sub_Intv_Id into ACCA_Acct.Sub_Intv_Id.
Place ACCA_Acct into the IDP_Acct store for this scene, identified
by the ACCA_Acct.Sub_Intv_Id.

Create a IDP_Status.IDP_ACCA_Status message indicating CCA scores have been generated for the scene.

Reusability

Possibilities for reuse include:

Generating accounting information. This function will be needed by almost every component of the LPS so there is a possibility the code could be reused.

Cloud Cover Assessment algorithm. A prototype was created for the ACCA. If the algorithm used in the prototype is the eventual assessment algorithm or if the algorithm used is from another project, much of the code is reusable. For more information on the prototype, please refer to the "Prototype White Paper", September 1994.

Generating and sending status messages. Every major function (and probably most minor functions) will need to send messages to the MACS.

4.8.2 Performance Requirements

The following list summarizes the performance requirements allocated to the IDPS:

4.8.1 The IDPS software on each LPS string shall provide the capability to process the equivalent of any combination of the format 1 and format 2 portions of 125 Landsat 7 ETM+ scenes of wideband data per day (approximately 25-30 GB per day).

4.8.2 The IDPS software on each LPS string shall process aligned band data at a minimum rate of not less than 7.5 Mbps (based on a minimum raw wideband throughput of 7.5 Mbps without PCD and CADU overhead).

4.8.3 The IDPS software on each LPS string shall provide the capability to execute concurrently with all other LPS subsystems.

4.8.3.1 The IDPS software on each LPS string shall begin to process received data immediately upon receipt of required inputs.

4.8.3.2 The IDPS software on each LPS string shall output scene metadata within 250 seconds of the time of receiving all required inputs.

4.8.4 The IDPS software on each LPS string shall provide the capability to reprocess a maximum of 10% of a string's daily input volume of data (approximately 12.5 scenes or 2.5-3 GB per day).

4.8.5 The IDPS software on each LPS string shall provide the capability to process received data at a daily average aggregate rate of 2.9 megabits per second (Mbps) (Includes 10% of overhead due to reprocessing).

4.8.6 The IDPS software on each LPS string shall maintain data processing throughput performance for all Landsat 7 raw data received with a BER of one bit error in 10^5 bits, without loss of level zero processed data and without retransmission.

4.8.7 The IDPS software on each LPS string shall output any periodic processing status information that it generates with a maximum latency of 30 seconds between outputs.

4.9 Management and Control Subsystem (MACS)

This subsystem is responsible for system level control and monitoring of LPS devices and processes, providing the interface between the LPS system and the operator, and generating the Metadata files on a subinterval basis.

4.9.1 Functional Requirements

The following list summarizes the functional requirements allocated to the MACS:

- provide an orderly system start-up and shut-down capability.
- provide the capability to monitor and control LPS operation, including the display of error messages or alarms to the operator.
- generate Level 0R metadata file(s) on a subinterval basis.
- provide the capability to display and/or print LPS summary or quality and accounting information upon operator request.
- provide monitoring test points and indicators to verify proper operation of system capabilities and components.
- provide the capability to manually override LPS automated functions or to suspend LPS file generation.
- notify LDTS when to send the DAN to LP DAAC.

4.9.1.1 Major Functions

Upon receipt of operator keyed Directive, the MACS first determines which subsystem the directive belongs to and forwards the Directive to the target subsystem. If the Directive is either MACS_Modify_Schedule_Drct or MACS_Modify_Config_Drct, it sends the Directive to Modify Contact Schedule or Modify LPS Configuration function to modify the appropriate data store. If the Directive is MACS_Control_Drct, it sends it to LPS System Control function for LPS system level control.

When the RDP_Process_Drct is issued, the Contact_Id is forwarded to the Generate Metadata process, which generates a metadata file for

each subinterval in the contact period. Once all associated metadata files are completed, the MACS sends a notification (LDT_Send_DAN) with Contact_Id to the LDTS for the readiness of file transfer to LP DAAC.

During LPS operation, the MACS monitors for system failure. It receives the processing status (RDC_Return_Status, RDP_Return_Status, MFP_Return_Status, PCD_Return_Status, IDPS_Return_Status, and LDTS_Return_Status) from the subsystems and displays it to the operator. It also displays or reports the quality and accounting information (LPS_Acct) or summary information (Data_Receive_Summary, Return_Link_QA_ Report, Level0R_QA, and LDT_Transfer_Sum) upon operator request.

The major functions of MACS are depicted in the following data flow diagrams.

Figure 4-18
MACS - DFD 6.0

4.9.1.2 Interface Requirements

The following two tables summarize the interface requirements for the MACS:

Table 4.11 MACS Interface Requirements - INPUT

Input Item Name	Source	Description
Current_Time	Time Source	System wide time source
Directive	Operator	Operator commands
RDC_Status	RDCS	Processing Status
RDP_Status	RDPS	Processing Status
MFP_Status	MFPS	Processing Status
PCD_Status	PCDS	Processing Status
IDP_Status	IDPS	Processing Status
LDT_Status	LDTS	Processing Status
Report_RDC_Data_Capture_Sum	RDCS	Data capture summary on a contact period basis
Report_RDP_Return_Link_QA	RDPS	Return_Link quality and accounting information report
Report_MFP_L0R_QA	MFPS	Level_0R file quality and accounting information report
Report_LDT_File_Xfer_Sum	LDTS	LPS file transfer summary report on a contact period basis
LPS_Acct	RDCS, RDPS, MFPS, PCDS, IDPS	LPS quality and accounting information for metadata file creation on a subinterval basis
Sub_Intv	MFPS	Subinterval information for metadata file creation
Valid_WRS_Parms	PCDS	Scene path/row and longiture/latitude information

Table 4.12 MACS Interface Requirements - OUTPUT

Output Item Name	Destination	Description
RDC_Directive	RDCS	Control Directive
RDP_Directive	RDPS	Control Directive
MFP_Directive	MFPS	Control Directive
PCD_Directive	PCDS	Control Directive
IDP_Directive	IDPS	Control Directive
LDT_Directive	LDTS	Control Directive
Contact_Schedules	RDCS	Data receive schedule
LPS_Configuration	RDCS, RDPS, MFPS, LDTS	LPS setup parameters
LPS_Report	Operator	LPS summary report
LPS_Status	Operator	LPS processing status
Metadata_File	LDTS	Metadata file on a subinterval basis
LDT_Send_DAN	LDTS	LPS files ready for transmission notification

4.9.1.3 Detailed Functional Requirements

This section contains the process specifications for the lowest level processes in the MACS data flow diagrams.

NAME:
6.1

TITLE:
Process LPS Directive

INPUT/OUTPUT:
Contact_Id : data_out
MACS_Modify_Schedule_Drct : data_out
MACS_Modify_Config_Drct : data_out
MACS_Directive_Dispatch_Status : data_out
MACS_Control_Drct : data_out
MFP_Directive : data_out
LDT_Directive : data_out
IDP_Directive : data_out
PCD_Directive : data_out
RDP_Directive : data_out
RDC_Directive : data_out
Directive : data_in

BODY:
Description of Process

Accept the Directive from LPS operator and send the Directive to the target subsystem(s) or the MACS internal functions.

Assumptions

Preconditions

None.

Post conditions

All Directives are sent to their appropriate destination.

Constraints

None.

Functional Breakdown

Accept the Directive keyed from LPS operator.

Forward the Directive to the appropriate subsystem(s) as shown in the following:

RDC_Directive to RDCS
RDP_Directive to RDPS
MFP_Directive to MFPS
PCD_Directive to PCDS
IPD_Directive to IDPS
LDT_Directive to LDTS
MACS_Modify_Schedule_Drct to Modify Contact Schedule Function
MACS_Modify_Conf_Drct to Modify LPS Configuration Function
MACS_Control_Drct to LPS System Control

When the RDP_Directive.RDP_Process_Drct is issued to RDPS, then
Send Contact_Id to the Generate Metadata process.

Send the directive dispatching status as
MACS_Directive_Dispatch_Status to Report LPS Status
function for display.

Reusability
The Oracle SQL*FORMS may be used to validate the inputs and route the directives.

NAME:
6.2;15

TITLE:
Generate Metadata

INPUT/OUTPUT:
Metadata_File : data_out
MACS_Metadata_Generation_Status : data_out
LDT_Send_DAN : data_out
Sub_Intv : data_in
LPS_Acct : data_in
Current_Time : data_in
Valid_WRS_Parms : data_in
LPS_Configuration : data_in
Contact_Id : data_in

BODY:
Description of Process
Create the Metadata_File from LPS subinterval based quality and accounting information upon receipt of the Sub_Intv information from the MFPS.

Assumption
Preconditions
The ACCA information and the quality and accounting files (LPS_Acct) associated with current processed sub-intervals are completed upon receipt of the Sub_Intv information from the MFPS.
The latitude and longitude information for corner quadrants is available from Valid_WRS_Parms.

Post conditions
The Metadata_File is created on a subinterval basis and the LPS_File_List associated with this subinterval is also generated.

Constraints
None

Functional Breakdown
Receive the Contact_Id.
For each Sub_Intv.Sub_Intv_Id identified by Contact_Id, do the following:

Use Contact_Id to identify the information within LPS_Acct.RDC_Acct necessary for the Metadata_File.
Use Contact_Id to identify the information within LPS_Acct.RDP_Acct necessary for the Metadata_File.
Use Sub_Intv.Sub_Intv_Id to identify the information within LPS_Acct.MFP_Acct necessary for the Metadata_File.
Use Sub_Intv.Sub_Intv_Id to identify the information within LPS_Acct.PCD_Acct necessary for the Metadata_File.
Use Sub_Intv.Sub_Intv_Id to identify the information within

LPS_Acct.IDP_Acct necessary for the Metadata_File.
Verify all of the quality and accounting stores in LPS_Acct are completed for the current processed Sub_Intv.Sub_Intv_Id.

Create the Metadata_File_Name using the LPS_Configuration, File_Version_Number and the Sub_Intv.MF_Start_Time identified by the current Sub_Intv_Id.
Create the Metadata_Header with the following information:
Current_Time,
LPS_Configuration.File_Version_Number,
Sub_Intv.Contact_Id.LPS_Hardware_String_Id,
LPS_Configuration.LPS_Software_Version_Number,
LPS_Configuration.Spacecraft_Id,
LPS_Acct.MFP_Acct.Mjf_VCDU_QA.VCDU_QA.ETM_Data_Format,
LPS_Configuration.Instrument_Id,
Sub_Intv.Contact_Id.Contact_Start_Time,
LPS_Acct.PCD_Acct.Orbit_Num,
Sub_Intv.MF_Start_Time,
Sub_Intv.MF_Stop_Time,
LPS_Acct.MFP_Acct..Mjf_VCDU_QA.Mjf_QA.Mjf_Count,
LPS_Acct.PCD_Acct.Num_PCD_MJF,
LPS_Acct.PCD_Acct.First_PCD_MJF_Time,
LPS_Acct.PCD_Acct.WRS_Path_Nominal,
LPS_Acct.PCD_Acct.WRS_Row_Nominal for first scene in subinterval determines Starting_Row,
LPS_Acct.PCD_Acct.WRS_Row_Nominal for last scene in subinterval determines Ending_Row,
LPS_Acct.PCD_Acct.PCD_File_Name,
LPS_Acct.IDP_Acct.Browse_File_Names,
LPS_Acct.MFP_Acct.Cal_File_Name,
LPS_Acct.MFP_Acct.MSCD_File_Name,
LPS_Acct.IDP_Acct.Band_File_Names,
LPS_Acct.IDP_Acct.Bands_Present, and
LPS_Acct.MFP_Acct.Mjf_VCDU_QA.VCDU_QA.ETM_Data_Format

Place the following information into the Metadata_File:

Metadata_Header,
LPS_Acct.MFP_Acct.Mjf_VCDU_QA.VCDU_QA.Mjf_CADU_BER_Cnt,
LPS_Acct.MFP_Acct.Mjf_VCDU_QA.VCDU_QA.Mjf_CADU_Sync_Err_Cnt,
LPS_Acct.MFP_Acct.Mjf_VCDU_QA.VCDU_QA.Mjf_CADU_Rcvd_Cnt,
LPS_Acct.MFP_Acct.Mjf_VCDU_QA.VCDU_QA.Mjf_CADU_Missing_Cnt,
Use the LPS_Acct.MFP_Acct.Mjf_VCDU_QA.VCDU_QA. Mjf_CADU_RS_Corr_Cnt, and the LPS_Acct.MFP_Acct. Mjf_VCDU_QA.VCDU_QA.Mjf_CADU_RS_Uncorr_Cnt to calculate VCDU_Hdr_Err_Count,
LPS_Acct.MFP_Acct.Mjf_VCDU_QA.VCDU_QA.Mjf_CADU_BCH_Corr_Cnt,
LPS_Acct.MFP_Acct.Mjf_VCDU_QA.VCDU_QA.Mjf_CADU_BCH_Uncorr_Cnt,
LPS_Acct.MFP_Acct.Mjf_VCDU_QA.VCDU_QA.Mjf_CADU_BCH_Bits_Corr,
LPS_Acct.MFP_Acct.Mjf_Full_Fill_Cnt,
LPS_Acct.MFP_Acct.Mjf_Part_Fill_Cnt,
LPS_Acct.MFP_Acct.Mjf_Time_Code_Err_Cnt,
LPS_Acct.PCD_Acct.Minor_Frame_Acct.Failed_PCD_Votes added for each major frame in the PCD cycle

will provide the Failed_PCD_Votes,
LPS_Acct.PCD_Acct.Minor_Frame_Acct.
Num_PCD_MNF_Sync_Errors added for each major
frame in the PCD cycle will provide the
Num_PCD_MNF_Sync_Errors,
LPS_Acct.PCD_Acct.Minor_Frame_Acct.Num_PCD_Filled_MNF
added for each major frame in the PCD cycle
will provide the Num_PCD_Filled_MNF,
LPS_Acct.PCD_Acct.Major_Frame_Acct.Num_PCD_Filled_MJF
added for each major frame in the PCD cycle
will provide the Num_PCD_Filled_MJF,
LPS_Acct.PCD_Acct.Major_Frame_Acct.Num_Avail_ADP,
LPS_Acct.PCD_Acct.Major_Frame_Acct.Num_Rejected_ADP,
LPS_Acct.PCD_Acct.Major_Frame_Acct.Num_Missing_ADP,
LPS_Acct.PCD_Acct.Major_Frame_Acct.Num_Avail_EDP,
LPS_Acct.PCD_Acct.Major_Frame_Acct.Num_Rejected_EDP,
LPS_Acct.PCD_Acct.Major_Frame_Acct.Num_Missing_EDP,
For each identified scene in LPS_Acct.PCD_Acct.
PCD_Scene_Count provide the following:

LPS_Acct.PCD_Acct.Sub_Intv_Scene_Num,
LPS_Acct.PCD_Acct.WRS_Path_Nominal,
LPS_Acct.PCD_Acct.WRS_Row_Nominal,
LPS_Acct.PCD_Acct.Scene_Center_Time,
LPS_Acct.PCD_Acct.Scene_Center_Scan_Num,
Valid_WRS_Parms.Center_Latitude,
Valid_WRS_Parms.Center_Longitude,
Valid_WRS_Parms.Upper_Left_Corner_Latitude,
Valid_WRS_Parms.Upper_Left_Corner_Longitude,
Valid_WRS_Parms.Upper_Right_Corner_Latitude,
Valid_WRS_Parms.Upper_Right_Corner_Longitude,
Valid_WRS_Parms.Lower_Left_Corner_Latitude,
Valid_WRS_Parms.Lower_Left_Corner_Longitude,
Valid_WRS_Parms.Lower_Right_Corner_Latitude,
Valid_WRS_Parms.Lower_Right_Corner_Longitude,
LPS_Acct.PCD_Acct.Sun_Azimuth,
LPS_Acct.PCD_Acct.Sun_Elevation,
LPS_Acct.IDP_Acct.ACCA.CCA_Quadrant1_Score,
LPS_Acct.IDP_Acct.ACCA.CCA_Quadrant2_Score,
LPS_Acct.IDP_Acct.ACCA.CCA_Quadrant3_Score,
LPS_Acct.IDP_Acct.ACCA.CCA_Quadrant4_Score,
LPS_Acct.IDP_Acct.ACCA.CCA_Aggregate_Score,
LPS_Acct.IDP_Acct.Gain_Change_Flag,
LPS_Acct.IDP_Acct.Band_Gain

Set LDT_Send_DAN.Contact_Id to Contact_Id
Set LDT_Send_DAN.Contact_File_Names.Sub_Intv_File_Names.
Metadata_File_Name to Metadata_File_Name
Set LDT_Send_DAN.Contact_File_Names.Sub_Intv_File_Names.
PCD_File_Name to LPS_Acct.PCD_Acct.PCD_File_Name.
Set LDT_Send_DAN.Contact_File_Names.Sub_Intv_File_Names.
Browse_File_Names to LPS_Acct.IDP_Acct.
Browse_File_Names.
Set LDT_Send_DAN.Contact_File_Names.Sub_Intv_File_Names.

Cal_File_Name to LPS_Acct.MFP_Acct.Cal_File_Name.
Set LDT_Send_DAN.Contact_File_Names.Sub_Intv_File_Names.
MSCD_File_Name to LPS_Acct.MFP_Acct.
Set LDT_Send_DAN.Contact_File_Names.Sub_Intv_File_Names.
Band_File_Names to LPS_Acct.IDP_Acct.Band_File_Names.
Output the LDT_Send_DAN notification to the LDTS if the current Sub_Intv_Id is the last subinterval of the currently processed contact period identified by Contact_Id.

Send the MACS_Metadata_Generation_Status to Report LPS Status function for display.

Reusability

The directive routing, status forwarding and message logging functions may be able to be reused from the existing projects.

NAME:
6.3;12

TITLE:
Report LPS Status

INPUT/OUTPUT:
LPS_Status : data_out
LPS_Journal : data_out
MACS_Directive_Dispatch_Status : data_in
MACS_Control_Status : data_in
PCD_Status : data_in
RDP_Status : data_in
MFP_Status : data_in
IDP_Status : data_in
LDT_Status : data_in
RDC_Status : data_in
MACS_Modify_Config_Status : data_in
MACS_Modify_Schedule_Status : data_in
MACS_Metadata_Generation_Status : data_in

BODY:
Description of Process
Report or display the LPS_Status to LPS operator.

Assumption
Preconditions
None.

Post conditions
The LPS_Status is displayed on the console to LPS operator.

Constraints
None.

Functional Breakdown
Receive incoming status messages from LPS Subsystems and other MACS functions.
Report the received status messages to the LPS operator as LPS_Status.
Log the LPS_Status message into the LPS_Journal system log.

Reusability
The LPS system monitoring function and the message logging function from PACOR II may be reusable.

NAME:
6.4;13

TITLE:
Display or Print LPS Report

INPUT/OUTPUT:
LPS_Report : data_out
Report_LDT_File_Xfer_Sum : data_in
Report_RDC_Data_Capture_Sum : data_in
Report_MFP_L0R_QA : data_in
Report_RDP_Return_Link_QA : data_in

BODY:
Description of Process

Display or generate a hardcopy report to LPS operator for the specified summary or quality and accounting information.

Assumption

Preconditions

The summary or quality and accounting information is provided by LPS subsystems prior to display or report.

Post conditions

The specified summary or quality and accounting information will be displayed on the console or printed in a hardcopy report.

Constraints

None.

Functional Breakdown

Receive

Report_RDC_Data_Capture_Sum,
Report_RDP_Return_Link_QA,
Report_MFP_L0R_QA,
Report_LDT_File_Xfer_Sum
from LPS subsystems.
Display or generate a hardcopy report for the received summary or quality and accounting information based on the report type specified in the received summary or quality and accounting information.

Reusability

This function may be satisfied using Database Report writer package.

NAME:
6.5;7

TITLE:
Modify LPS Configuration

INPUT/OUTPUT:
MACS_Modify_Config_Drct : data_inout
MACS_Modify_Config_Status : data_out
MACS_Modify_Config_Drct : data_in

BODY:
Description of Process
Modify information in the LPS_Configuration store.

Assumption
Preconditions
None.

Post conditions
None.

Constraints
None.

Functional Breakdown
Verify the MACS_Modify_Config_Drct input and
make sure the individual field format and range of the
entry are valid.
Replace the existing LPS_Configuration entry.
Output the MACS_Modify_Config_Status.

Reusability
This process can be satisfied with Oracle SQL.

NAME:
6.6;9

TITLE:
Modify Contact Schedule

INPUT/OUTPUT:
Contact_Schedules : data_inout
MACS_Modify_Schedule_Status : data_out
MACS_Modify_Schedule_Drct : data_in

BODY:
Description of Process
Modify information in the Contact_Schedules store.

Assumption
Preconditions
None.

Post conditions
None.

Constraints
None.

Functional Breakdown
Verify the MACS_Modify_Schedule_Drct input and
make sure the individual field format and range of the
entry are valid.
If we are modifying existing Contact_Schedules information, then
Replace the old Contact_Schedules information with the
new Contact_Schedules information identified by
MACS_Modify_Schedule_Drct.Contact_Schedule_Id.
If we are adding new Contact_Schedules information, then
Insert the new Contact_Schedules information into
the Contact_Schedules store assigning a new
MACS_Modify_Schedule_Drct.Contact_Schedule_Id.
If we are removing existing Contact_Schedules information, then
Remove the existing Contact_Schedules information from
the Contact_Schedules store identified by
MACS_Modify_Schedule_Drct.Contact_Schedule_Id.
Output the MACS_Modify_Schedule_Status.

Reusability
This process can be satisfied with Oracle SQL.

NAME:
6.7;6

TITLE:
LPS System Control

INPUT/OUTPUT:
MACS_Control_Status : data_out
MACS_Control_Drct : data_in

BODY:
Description of Process
Perform LPS system level control activity.

Assumption
Preconditions
None.

Post conditions
None.

Constraints
None.

Functional Breakdown
Bring up the LPS system if MACS_Control_Drct is a Startup.
Shut down the LPS system if MACS_Control_Drct is a Shutdown.

Assign MACS_Control_Status the result of any action taken

Reusability
None.

NAME:
6.8;3

TITLE:
Monitor System Faults

INPUT/OUTPUT:
LPS_Status : data_out
LPS_Journal : data_in

BODY:
Description of Process
Allows the operator access to the Activity log for LPS.

Assumptions
Preconditions
None.

Postconditions
None.

Constraints
None.

Functional Breakdown
Display the contents of LPS_Journal as LPS_Status.

Reusability
This function can be performed by a text editor.

4.9.2 Performance Requirements

The following list summarizes the performance requirements allocated to the MACS:

4.9.2.1 The MACS software on each LPS string shall forward any directive to start/stop data capture or to generate a data receive summary to the RDCS within one second of its receipt from the operator.

4.9.2.2 The MACS software on each LPS string shall display a data receive summary for the most recently received raw wideband data within one second of its receipt from the RDCS.

4.9.2.3 The MACS software on each LPS string shall submit a data receive summary for the most recently received raw wideband data to a print queue within 1 second of its receipt from the RDCS.

4.9.2.4 The MACS software on each LPS string shall provide the capability to execute concurrently with all other LPS subsystems.

4.9.2.4.1 The MACS software on each LPS string shall begin to process metadata immediately upon receipt of required inputs.

4.9.2.4.2 The MACS software on each LPS string shall output a metadata file within 240 seconds of the time of receiving all required inputs.

4.9.2.5 The MACS software on each LPS string shall maintain data processing throughput performance for all Landsat 7 raw wideband data received with a BER of one bit error in 10^5 bits, without loss of level zero processed data and without retransmission.

4.9.2.6 The mean time to bring up the MACS software on any LPS string (from operating system boot to readiness to accept operator inputs) shall not exceed 12 minutes (based on a 15 minute estimate from RMA analysis and allowing 3 minutes for operator initiation and network latencies, 5 minutes for OS boot, 5 minutes for DBMS startup, and 2 minutes for LPS software start-up).

4.9.2.7 The time to bring up the MACS software on any LPS string (from operating system boot to readiness to accept

operator inputs) shall not exceed twice the required mean time to bring up MACS software in 99 percent of all cases.

4.9.2.8 The MACS software on each LPS string shall output any periodic processing status information that it generates with a maximum latency of 30 seconds between outputs.

4.10 LPS Data Transfer Subsystem (LDTS)

This subsystem is responsible for sending a Data Availability Notification (DAN) to the LP DAAC about the availability of LPS files on a contact basis. Upon receiving a Data Transfer Acknowledgment (DTA), this subsystem is responsible for deleting the successfully transferred files. This subsystem is also responsible for generating a daily file transfer summary report upon request.

4.10.1 Functional Requirements

The following list summarizes the functional requirements allocated to the LDTS:

- Interface with the LP DAAC to coordinate the transfer of LPS output files to the LP DAAC.
- notify LP DAAC on the availability of LPS files.
- provide the capability to receive notification from LP DAAC on the successful receipt of transferred LPS files.
- provide the capability to store LPS data files until confirmation of successful transfer is received from the LP DAAC.
- provide a manual over-ride and protected capability to delete all LPS files on a specific contact period basis.
- provide a manual over-ride and protected capability to retain all LPS files on-line on a specific contact period basis.
- provide the capability to generate LPS file(s) transfer summary on a daily basis.
- provide the capability to manually override the LPS automated functions.
- provide the capability to selectively enable or disable the Transfer LPS Files function.

4.10.1.1 Major Functions

The Generate DAN function receives the LDT_Send_DAN directive from the MACS and then retrieves information from the LDT_Output_File_Info data store to build a data availability notification (DAN).

The DAN is passed to the Send DAN function which sends it to the LP DAAC. Sending DAN can be enabled or disabled by operator via the Control Send DAN function. The DAN will be resent upon operator's request (LDT_Resend_DAN_Drct).

The Receive DTA function processes the DTA received from the LP DAAC and notifies the Delete LPS Files function to delete successfully transferred files. File deletions can be overridden by operator via the Retain LPS File function.

The LDT_Output_File_Info data store is maintained by the Generate DAN, Delete LPS Files, Retain LPS Files and Receive DAT functions. It is used by the Generate Transfer Summary Report function to build and send to MACS a Report_LDT_File_Xfer_Sum.

The Transfer Files function handles file transfers between the LPS and LP DAAC.

The major functions of LDTS are depicted in the following data flow diagram.

Figure 4-19
LDTS - DFD 7.0

4.10.1.2 Interface Requirements

The following two tables summarize the interface requirements for the LDTS:

Table 4.13 LDTS Interface Requirements - INPUT

Input Item Name	Source	Description
Current_Time	Time Source	System wide time source
LDT_Send_DAN	MACS	Subinterval files complete notice
LDT_Directive	MACS	Control directives from operator.
Metadata_File	MACS	It contains LPS quality and accounting information.
MSCD_File	MFPS	It contains mirror scan correction data.
Cal_File	MFPS	It contains calibration data for each major frame.
Band_File	IDPS	A set of files of de-interleaved data with one file per band.
Browse_File	IDPS	It contains reduced data-volume image files.
PCD_File	PCDS	It contains PCD major frames.
Transfer_Request	LP DAAC	Data transfer request
DTA	LP DAAC	Data transfer acknowledgment

Table 4.14 LDTS Interface Requirements - OUTPUT

Output Item Name	Destination	Description
Report_LDT_File_ Xfer_Sum	MACS	A summary report of LPS file transfers.
DAN	LP DAAC	Data available notification sent to LP DAAC from LPS.
LDT_DAN_Status	MACS	It indicates either a DAN has been sent or communication problems.
LDT_DTA_Status	MACS	It indicates a DTA has been received from LP DAAC.
LDT_Retain_Files_ Status	MACS	It indicates success or failure of an LDT_Retain_Files_Drct.
LDT_Delete_Files_ Status	MACS	It indicates success or failure of an LDT_Delete_Files_Drct.
LPS_Files	LP DAAC	The LPS level 0R files transferred to the LP DAAC via file transfer protocols.
LPS_File_Transfer_ Status	LP DAAC	The file transfer status generated from the file transfer protocols.

4.10.1.3 Detailed Functional Requirements

This section contains the process specifications for the lowest level processes in the LDTS data flow diagrams.

NAME:
7.1;10

TITLE:
Generate DAN

INPUT/OUTPUT:
Time_Available : data_out
Internal_DAN : data_out
Contact_Id : data_out
Contact_File_Names : data_out
LDT_Send_DAN : data_in
Current_Time : data_in

BODY:
Description of Process

This function constructs and sends a Data Availability Notice (DAN) to the LDTS "Send DAN" function.

Assumptions

Preconditions

None.

Post conditions

None.

Constraints

None.

Functional Breakdown

Build an Internal_DAN:

Set Internal_DAN.Contact_Id to LDT_Send_DAN.Contact_Id
Set Internal_DAN.Contact_File_Names to
LDT_Send_DAN.Contact_File_Names

Create an LPS_Output_File_Info entry

Set LPS_Output_File_Info.Contact_Id to LDT_Send_DAN.Contact_Id
Set LPS_Output_File_Info.Contact_File_Names to
LDT_Send_DAN.Contact_File_Names.
Set LPS_Output_File_Info.Time_Available to Current_Time.

Send Internal_DAN to "Send DAN" function.

Reusability

Reuse of DDF and Pacor II's DAN functionality is a possibility.

NAME:
7.2

TITLE:
Send DAN

INPUT/OUTPUT:
DAN_Suspended : data_inout
LDT_DAN_Status : data_out
DAN_Transmission_Time : data_out
DAN : data_out
LDT_Resend_DAN_Drct : data_in
DAN_Transfer_State : data_in
Current_Time : data_in
Contact_File_Names : data_in
Internal_DAN : data_in

BODY:
Description of Process
This function sends or resends a data availability notifications (DANs) to the LP DAAC. If sending DANs is disabled, the DAN is stored for sending once sending is enabled.

Assumptions
Preconditions
None
Post conditions
None
Constraints
None

Functional Breakdown
If DAN_Transfer_State is "ENABLED":
If LDT_Resend_DAN_Drct is received:
Retrieve LDT_Output_File_Info
where LDT_Output_File_Info.Contact_Id equals
LDT_Resend_DAN_Drct.Contact_Id
Set DAN.Contact_Id to LDT_Resend_DAN_Drct.Contact.Id
Set DAN.Contact_File_Names to
LDT_Output_File_Info.Contact_File_Names
If LDT_Output_File_Info.DAN_Suspended is "SUSPENDED" then
Update LDT_Output_File_Info.DAN_Suspended to
"PROCESS".

If Internal_DAN is received:
Retrieve LDT_Output_File_Info
where LDT_Output_File_Info.Contact_Id equals
Internal_DAN.Contact_Id
Set DAN.Contact_Id to Internal_DAN.Contact_Id
Set DAN.Contact_File_Names to Internal_DAN.Contact_File_Names

Send the DAN to the LP DAAC

Append Current_Time to LDT_Output_File_Info.DAN_Transmission_Time

list.

Send LDT_DAN_Status to MACS indicating the Contact_Id and status of the DAN that was sent.

If DAN_Transfer_State is "DISABLED":
If Internal_DAN is received:
Retrieve LDT_Output_File_Info
where LDT_Output_File_Info.Contact_Id equals Internal_DAN.Contact_Id
If LDT_Resend_DAN_Drct is received:
Retrieve LDT_Output_File_Info
where LDT_Output_File.Contact_Id equals LDT_Resend_DAN.Drct.Contact_Id.
Update LDT_Output_File_Info.DAN_Suspended to true
Send LDT_DAN_Status to MACS indicating that DAN transfer is disabled and the DAN associated with Contact_Id was not sent

Reusability
Possible reuse of Pacor II/DDF code depending on the ICD between LPS and LP DAAC.

NAME:
7.3;10

TITLE:
Receive DTA

INPUT/OUTPUT:
Contact_Id : data_out
LDT_DTA_Status : data_out
DTA_Time_Of_Receipt : data_out
Current_Time : data_in
DTA : data_in

BODY:
Description of Process

This function extracts file transfer information from the input data transfer acknowledgment (DTA). It notifies MACS that a DTA was received and updates LPS_Output_File_Info with the receipt time if the DTA indicates successful file transfer.

Assumptions

Preconditions

None.

Post conditions

LPS_Output_File_Info data store will be updated with the time of receipt of the DTA if DTA indicates success.

Constraints

DTA format will be defined in the ICD between LPS and LP DAAC.

Functional Breakdown

DTA indicates successful transfer of LPS files to LP DAAC:

Append Current_Time to LDT_Output_File_Info.DTA_Time_Of_Receipt for LDT_Output_File_Info.Contact_Id equal to DTA.Contact_Id

Send Contact_Id to "Delete LPS Files" function

Send MACS LDT_DTA_Status indicating successful transfer of files for contact DTA.Contact_Id

DTA indicates unsuccessful transfer of LPS files to LP DAAC:

Send MACS LDT_DTA_Status indicating unsuccessful transfer of files for contact DTA.Contact_Id

Reusability

None.

NAME:
7.4;7

TITLE:
Transfer Files

INPUT/OUTPUT:
LPS_File_Transfer_Status : data_out
LPS_Files : data_out
Transfer_Request : data_in
Metadata_File : data_in
PCD_File : data_in
Browse_File : data_in
Band_File : data_in
Cal_File : data_in
MSCD_File : data_in

BODY:
Description of Process
This function transfers LPS Output files to the LP DAAC in response to a request from LP DAAC.

Assumptions
Preconditions
None.

Postconditions
None.

Constraints
A COTS File transfer protocol (as per the ICD between LPS and LP DAAC) will be used.

Functional Breakdown
Send LPS_Files (consisting of Band_File, PCD_File, Browse_File, Cal_File, MSCD_File, and Metadata_File to the LP DAAC in response to receipt of a Transfer_Request.
Send LPS_File_Transfer_Status, indicating the status of the requested file transfer operations, to the LP DAAC.

Reusability
COTS software will be used.

NAME:
7.5;10

TITLE:
Generate Transfer Summary Report

INPUT/OUTPUT:
Report_LDT_File_Xfer_Sum : data_out
LDT_Rpt_File_Xfer_Sum_Drct : data_in
Current_Time : data_in
LDT_Output_File_Info : data_in

BODY:
Description of Process
This function generates the LPS file transfer summary report upon receiving a request from the MACS.

Assumptions
Preconditions
None.

Post conditions
None.

Constraints
None.

Functional Breakdown
Calculate Report_Begin_Time equal to midnight of the day represented by LDT_Rpt_File_Xfer_Sum_Drct.Date
Calculate Report_End_Time equal the point in time which is the earlier of
1) Current_Time
2) Midnight of the day following LDT_Rpt_File_Xfer_Sum_Drct.Date

Set Report_LDT_File_Xfer_Sum.Current_Time equal to Current_Time
Set Report_LDT_File_Xfer_Sum.Date_Of_Report_Data to Report_LDT_File_Xfer_Sum.Date

Set Report_LDT_File_Xfer_Sum.Available_File_Names to all LDT_Output_File_Info.Contact_File_Names where
LDT_Output_File_Info.Time_Available is less than or equal to Report_End_Time,
There are no times in the LDT_Output_File_Info. DAN_Transmission_Time list less than or equal to Report_End_Time, and any value in LDT_Output_File_Info.Time_Deleted is greater than Report_End_Time.

Set Report_LDT_File_Xfer_Sum.Available_L0R_File_Count equal to the number of L0R files in Report_LDT_File_Xfer_Sum.Available_File_Names
Set Report_LDT_File_Xfer_Sum.Available_Metadata_File_Count equal to the number of Metadata files in Report_LDT_File_Xfer_Sum.Available_File_Names
Set Report_LDT_File_Xfer_Sum.Available_Browse_File_Count equal to the number

of Browse files in Report_LDT_File_Xfer_Sum.Available_File_Names

Set Report_LDT_File_Xfer_Sum.Online_File_Names to
all LDT_Output_File_Info.Contact_File_Names where
There is at least one time in the
LDT_Output_File_Info.DAN_Transmission_Time list less than or equal to Report_End_Time, and there are no times in the LDT_Output_File_Info.
DTA_Time_Of_Receipt list less than or equal to Report_End_Time, and any value in LDT_Output_File_Info.Time_Deleted is greater than Report_End_Time.

Set Report_LDT_File_Xfer_Sum.Online_L0R_File_Count equal to the number of L0R files in Report_LDT_File_Xfer_Sum.Online_File_Names
Set Report_LDT_File_Xfer_Sum.Online_Metadata_File_Count equal to the number of Metadata files in Report_LDT_File_Xfer_Sum.
Online_File_Names
Set Report_LDT_File_Xfer_Sum.Online_Browse_File_Count equal to the number of Browse files in Report_LDT_File_Xfer_Sum.Online_File_Names

Set Report_LDT_File_Xfer_Sum.Transmitted_File_Names to
all LDT_Output_File_Info.Contact_File_Names where
LDT_Output_File_Info.DTA_Time_Of_Receipt is greater than or equal to Report_Start_Time and less than Report_End_Time.

Calculate Report_LDT_File_Xfer_Sum.Volume_Of_Retained_Data
Calculate Report_LDT_File_Xfer_Sum.Available_Retention_Space

Send Report_LDT_File_Xfer_Sum to the MACS.

Reusability
System Software will be used to calculate Volume_Of_Retained_Data and Available_Retention_Space.

NAME:
7.6;14

TITLE:
Delete LPS Files

INPUT/OUTPUT:
Time_Deleted : data_out
LDT_Delete_Files_Status : data_out
LDT_Delete_Files_Drct : data_in
Current_Time : data_in
Metadata_File : data_in
PCD_File : data_in
Browse_File : data_in
Band_File : data_in
Cal_File : data_in
MSCD_File : data_in
Marked_For_Retention : data_in
Contact_Id : data_in

BODY:
Description of Process
This function deletes LPS files for a contact period that has not been marked for retention. The entry in LDT_Output_File_Info corresponding with the Contact_Id is time stamped with the time the Contact_Id's files were deleted.

Assumptions
Preconditions
None.

Post conditions
LPS files for the given Contact_Id have been deleted.
LDT_File_Info is updated with the time of deletion.

Constraints
None.

Functional Breakdown
If LDT_Delete_Files_Drct message received from MACS:
(this means override any file retention previously specified)
For each file in Metadata_File, PCD_File, Cal_File, MSCD_File, Browse_File, and Band_File associated with LDT_Delete_Files_Drct.Contact_Id :
Delete the file from LPS storage.
Set LDT_Output_File_Info.Time_Deleted to Current_Time for LDT_Output_File_Info.Contact_Id equal to Contact_Id

If Contact_Id from "Receive DTA" function is received
Retrieve LDT_Output_File_Info.Marked_For_Retention for LPS_Output_File_Info.Contact_Id equal to Contact_Id
If Marked_For_Retention indicates "DELETE" then
For each file in Metadata_File, PCD_File, Cal_File, MSCD_File,

Browse_File, and Band_File associated with Contact_Id

Delete the file from LPS storage.

Set LDT_Output_File_Info.Time_Deleted to Current_Time for LDT_Output_File_Info.Contact_Id equal to Contact_Id

Send a status message containing Contact_Id to MACS in LDT_Delete_Files_Status .

Reusability

None.

NAME:
7.7;11

TITLE:
Retain LPS Files

INPUT/OUTPUT:
Marked_For_Retention : data_out
LDT_Retain_Files_Status : data_out
LDT_Retain_Files_Drct : data_in

BODY:
Description of Process

This function marks LPS output files associated with a Contact_Id for retention. The files must then be manually deleted or deleted by a delete directive from the MACS.

Assumptions

Preconditions

None.

Post conditions

None.

Constraints

None.

Functional Breakdown

If there exists an entry in LDT_Output_File for LDT_Retain_Files_Drct.Contact_Id and it does not have a value for Time_Deleted:

Retrieve LPS_Output_File_Info where LPS_Output_File_Info.Contact_Id is equal to LDT_Retain_Files_Drct.Contact_Id.

Update LDT_Output_File_Info.Marked_For_Retention to "RETAIN"

Send a status message containing Contact_Id as LDT_Retain_Files_Status to the MACS

There is no entry in LDT_Output_File_Info for LDT_Retain_File_Drct.Contact_Id or there is one but it has a value for Time_Deleted:

Send LDT_Retain_Files_Status to the MACS containing LDT_Retain_Files_Drct.Contact_Id and an indication that the files were not found.

Reusability

None.

NAME:
7.8;9

TITLE:
Control Send DAN

INPUT/OUTPUT:
LDT_Resend_DAN_Drct : data_out
DAN_Transfer_State : data_out
LDT_Enable_File_Xfer_Drct : data_in
LDT_Disable_File_Xfer_Drct : data_in
DAN_Suspended : data_in
Contact_Id : data_in

BODY:
Description of Process
This function maintains a flag which indicates whether or not LPS DAN transfer is enabled.

Assumptions
Preconditions
None.

Postconditions
None.

Constraints
None.

Functional Breakdown
LDT_Enable_File_Xfer_Drct is received:
Update DAN_Transfer_State to indicate that LPS DAN transfer is enabled.
For each Contact_Id in LPS_Output_File_Info
where LDT_Output_File_Info.DAN_Suspended is "SUSPENDED"
Set LDT_Resend_DAN_Drct.Contact_Id to Contact_Id
Send LDT_Resend_DAN_Drct to LDTS "Send DAN"

LDT_Disable_File_Xfer_Drct is received:
Update DAN_Transfer_State to indicate that LPS DAN transfer is disabled.

Reusability
None.

4.10.2 Performance Requirements

The following list summarizes the performance requirements allocated to the LDTS:

4.10.2.1 The LDTS software on each LPS string shall provide the capability to transfer the equivalent of any combination of the format 1 and format 2 portions of 125 Landsat 7 ETM+ scenes of wideband data per day (approximately 25-30 GB per day). [4.1.3]

4.10.2.2 The LDTS software on each LPS string shall provide the capability to execute concurrently with all other LPS subsystems.

4.10.2.2.1 The LDTS software on each LPS string shall output a DAN within 240 seconds of the time of receiving all required inputs.

4.10.2.3 The LDTS software on each LPS string shall provide the capability to reprocess a maximum of 10% of a string's daily input volume of wideband data (approximately 12.5 scenes or 2.5-3 GB per day).

4.10.2.4 The LDTS software on each LPS string shall maintain data processing throughput performance for all Landsat 7 raw wideband data received with a BER of one bit error in 10^5 bits, without loss of level zero processed data and without retransmission.

4.10.2.5 The LDTS software on each LPS string shall provide the capability to transfer the string's daily volume of LPS output files to the LP DAAC at an average aggregate rate of 10 Mbps.

4.10.2.6 The LDTS software on each LPS string shall output any periodic processing status information that it generates with a maximum latency of 30 seconds between outputs.

5.0 Database Analysis

This section presents the results of LPS database design during the software requirement definition phase. The requirement definition phase includes requirement collection and analysis, conceptual design, and preliminary logical design as described in the following paragraphs. The LPS database design follows the standard database methodology which encompasses five major steps as illustrated in Figure 5.1. The methodology is consistent with SEAS System Development Methodology (SSDM). The design process consists of two parallel activities at each step: (1) Data view - the design of data content and structure of the database; (2) Process view - the design of processing and software application. Both views are described for each step of the process.

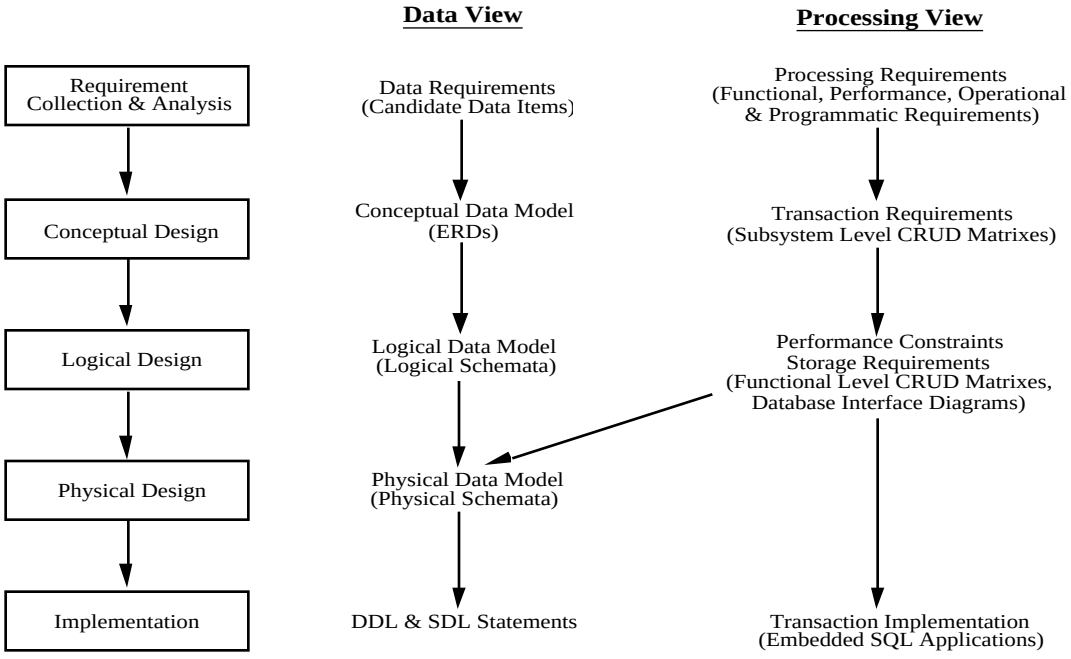


Figure 5.1 LPS Database Design Approach

- Requirement Collection & Analysis - This step is to identify and analyze the intended uses of the database. From the data point of view, this step identifies the LPS information to be stored in the database. From the processing perspective, this step analyzes the functional, performance, operational, and programmatic requirements of the LPS database. This step has been completed during the software requirement definition phase.

- Conceptual Design - This step is to create a high level data model which is Database Management System (DBMS) independent. From the data point of view, this step examines the data requirements resulting from the previous step and produces a conceptual database model which consists of Entity Relationship Diagrams (ERDs). From the processing perspective, this step examines the interaction between LPS subsystems and the database, and produces subsystem level Create, Retrieve, Update, and Delete (CRUD) matrices. This step has been completed during the software requirement definition phase.

- Logical Design - This step is to create a logical data model for a relational DBMS. From the data point of view, this step creates logical schemata and constraints to represent entities and relationships from the ERDs. Principles of normalization are applied to create well-structured schemata. From the processing perspective, this step produces functional CRUD matrices and database interface diagrams that describe the interaction between main processes within a subsystem and the database. This step also performs a preliminary analysis of data usage, performance constraints, and storage requirements. Part of this step has been completed during the software requirement definition phase while the rest will be continued in the next phase.

- Physical Design - This step is to select specific storage structures and access methods for the database. From the data point of view, this step determines storage structures such as tables, views, indices, and organizations of database information. From the processing perspective, this step further analyzes data usage and performance constraints which leads to the determination of access methods.

- Implementation - This step implements the database. From the data point of view, this step physically creates the LPS databases, tables, views, and indices through Data Definition Language (DDL) and Storage Definition Language (SDL) statements. From the processing perspective, this step implements database applications. Database transactions are examined and corresponding program code with embedded Structured Query Language (SQL) commands are written and

tested. Once the transactions are ready, the database will be populated.

The Cadre Teamwork Information Modeling (IM) tool is used during the design process. Teamwork/IM is an integrated toolset that helps database engineers model the entities, relationships, and attributes of all LPS data at the conceptual level. The tool is used to:

- Create an LPS conceptual model consisting of entity relationship diagrams.
- Ensure the completeness and consistency of database design.
- Generate code to create database tables and to enforce integrity constraints.
- Support documentation production.
- Simplify the maintenance effort by easing impact analysis and change implementation.
- Enhance configuration management by maintaining baselines.

5.1 Requirement Analysis and Conceptual Design

The purpose of database requirements analysis is to identify and analyze the intended uses of the database. This includes the identification of the information to be stored in the database and the analysis of the functional, performance, operational, and programmatic requirements of the database. Conceptual design creates a high level data model which consists of ERDs and examines the interaction between LPS subsystems and the database. This section presents the results of the requirement analysis and conceptual design phase.

5.1.1 Functional Requirement Analysis

This section presents a functional overview of the information tracking requirements for the LPS. The functions are described in terms of their interactions with the database. The functions are based on the LPS subsystems and functions presented in section 3 of Landsat 7 Processing System (LPS) System Design Specification (SDS).

5.1.1.1 Raw Data Capture Subsystem (RDCS)

The Raw Data Capture Subsystem (RDCS) captures and manages raw wideband data received from the LGS. Each LGS channel's data stream for a contact period is captured to the on-line disk. The data set is subsequently copied to removable media. Upon request, the RDCS retrieves a raw wideband data set from removable media and returns it to the on-line storage for reprocessing. On request, the RDCS generates a data receive summary for a specified raw wideband data set. The interactions between the RDCS and the LPS database are as follows:

- The RDCS retrieves the LPS configuration parameters such as the LPS hardware string identification and the LGS channel associated with the LPS string from the database when the RDCS starts up.

- The RDCS captures a raw wideband data bit stream for a single channel from the LGS and outputs the stream as a byte stream data set to an on-line raw wideband data store. At the end of the contact period, the subsystem collects and stores a contact summary describing the contact in the database.

- Upon request, the RDCS queries the database and generates a contact data receive summary for displaying and printing.

- Upon request, the RDCS retrieves the requested raw wideband data from removable media to the on-line storage for reprocessing. If not already present, contact summary information is collected and stored in the database.

5.1.1.2 Raw Data Processing Subsystem (RDPS)

The Raw Data Processing Subsystem (RDPS) inputs the rate buffered wideband data from the on-line data store and performs CCSDS Advanced Orbiting Systems (AOS) Grade 3 service on the received Channel Access Data Units (CADUs). This subsystem performs frame synchronization and Pseudo-Random Noise (PN) decoding of the received CADUs, Cyclic Redundancy Check (CRC) on VCDUs, and Reed-Solomon (RS) error detection and correction of VCDU headers. It also performs the BCH error detection and correction processing. The RDPS collects and generates raw data processing accounting and subsystem status information. The interactions between the RDPS and the LPS database are as follows:

- The RDPS receives CCSDS processing parameters and thresholds, transforms them into the subsystem's internal format (if necessary), and stores them in the database.

- The RDPS retrieves CCSDS parameters and thresholds for the frame synchronization processing. In addition, thresholds are also retrieved and used during the CCSDS AOS Grade 3 service and BCH processing.

- The RDPS processes the raw wideband data set for a contact period. It uses the CCSDS parameters and thresholds retrieved from the database for its processing and error reporting. The RDPS accumulates and collects quality and accounting information on a contact basis. The collected quality and accounting information is then stored in the database.

- Upon request, the RDPS queries the database and generates return link quality and accounting summary for displaying and printing.

5.1.1.3 Major Frame Processing Subsystem (MFPS)

The Major Frame Processing Subsystem (MFPS) processes annotated CADUs. This subsystem provides the functionality to synchronize the major frames, extract major frame times, deinterleave band data, reverse band data if necessary, and align band data. It is also responsible for generating the Calibration and Mirror Scan Correction files. In addition, it determines subinterval boundaries and extracts and provides PCD information to the PCDS subsystem. The MFPS also collects and generates Level 0R accounting and subsystem status information. The interactions between the MFPS and the LPS database are as follows:

- The MFPS receives, validates, and stores the sensor alignment information, the subinterval threshold, VCDU processing thresholds, major frame processing thresholds, and scan line processing thresholds in the database.

- The MFPS retrieves and uses the subinterval threshold to determine subintervals. Subintervals generated are stored in the database which are used by other subsystems.

- The MFPS retrieves and uses thresholds for error reporting during its processing.

- The MFPS retrieves and uses sensor alignment information for band alignment processing.

- The MFPS collects Level 0R quality and accounting information on a subinterval basis. Information collected is stored in the database which is used by the MACS for metadata generation.

- Upon request, the MFPS queries the database and generates Level 0R quality and accounting information for displaying and printing.

5.1.1.4 PCD Processing Subsystem (PCDS)

The PCD Processing Subsystem (PCDS) processes the Payload Correction Data (PCD). This subsystem is responsible for performing PCD byte majority voting, PCD major frame building, and PCD file generation. In addition, it identifies scenes using the Worldwide Reference System (WRS), calculates sun azimuth and elevation information, and extracts ETM+ calibration door events. The PCDS also collects and generates PCD accounting and subsystem status information. The interactions between the PCDS and the LPS database are as follows:

- The PCDS receives, validates, and stores the frame fill values, frame quality parameters, error report thresholds, and scene calculation parameters in the database. In addition, the WRS scene information is also stored in the database.

- The PCD retrieves and uses the frame fill values and frame quality parameters during the process of assembling PCD cycles.

- The PCD retrieves and uses error report thresholds for error reporting during its processing.

- The PCD retrieves and uses scene calculation parameters and the WRS information to identify WRS scenes and calculate sun azimuth and elevation. Calculated scene and sun information is stored in the database and used by other subsystems.

- The PCDS collects quality and accounting information during its processing of PCD data. Collected quality and accounting information is stored in the database and used by the MACS for metadata generation.

5.1.1.5 Image Data Processing Subsystem (IDPS)

The Image Data Processing Subsystem (IDPS) performs general LPS image data processing. The IDPS generates Level 0R instrument data files and browse files. It is also responsible for determining cloud coverage on a scene quadrant and full scene basis. The IDPS also collects and generates IDPS accounting and subsystem status information. The interactions between the IDPS and the LPS database are as follows:

- The IDPS receives, validates, and stores band parameters for browse and Automatic Cloud Cover Assessment (ACCA) in the database.
- The IDPS retrieves and uses subinterval information and band parameters to generate the browse files.
- The IDPS retrieves and uses subinterval information to generate the band files.
- The IDPS retrieves and uses subinterval information and band parameters for its ACCA processing. ACCA scores are generated and stored in the database to be included in the metadata file.
- The IDPS collects quality and accounting information during its processing. The collected information is stored in the database and used by the MACS for metadata generation.

5.1.1.6 Management and Control Subsystem (MACS)

The Management and Control Subsystem (MACS) provides an interface through which operations personnel can control the system operations, monitor overall system performance, access system accounting information, and manage the configuration parameters and thresholds which are used to drive system processing. The MACS also provides capabilities to generate metadata files and accounting information. The interactions between the MACS and the LPS database are as follows:

- The MACS manages the LPS system configuration parameters in the database. The subsystem allows the operator to update the LPS system configuration parameters.
- The MACS allows the operator to update the contact schedules.

- The MACS is responsible for generating metadata files. During the metadata file generation process, the MACS retrieves system configuration parameters, Level 0R quality and accounting information, PCD quality and accounting information, browse and band file accounting information, ACCA scores, WRS scene information, and subinterval data to generate metadata files on a subinterval basis. Metadata accounting and LPS file information is also generated and stored in the database for subsequent processing.

5.1.1.7 LPS Data Transfer Subsystem (LDTS)

The LPS Data Transfer Subsystem (LDTS) coordinates the transfer of LPS files to the LP DAAC over supported network connections. The LDTS is also responsible for overseeing the LPS file transmission, maintaining the status of all LPS files, managing the LPS output data store, and generating a file transfer summary. The interactions between the LDTS and the LPS database are as follows:

- The LDTS retrieves and uses configuration parameters and LPS file information from the database for the Data Availability Notice (DAN) generation. The LPS file information is organized on a contact basis.

- The LDTS maintains the status of all LPS files in the database. The status of LPS files is updated frequently to reflect Data Transfer Acknowledgments (DTAs) from the LP DAAC and directives from the MACS.

- The LDTS is responsible for managing the LPS output data storage. It retrieves the status of LPS files from the database when deciding whether to retain or delete certain LPS files.

- The LDTS retrieves and uses the LPS file information in the database to generate file transfer summaries.

5.1.2 Performance Requirement Analysis

Since all of the LPS subsystems interact with the database, the performance requirements for LPS subsystems include database performance considerations. Main factors affecting the database performance are discussed in the following subsections.

5.1.2.1 Response Time

The performance of the database is a part of the LPS performance. To provide the capability to process received wideband data at a minimum rate of 7.5 Mbps, the database access time must be minimized and optimized. To achieve this the LPS database must support the following capabilities:

- Storage Optimization - Distribution of database tables over different disk drives where speed of disk reads and writes is crucial.
- Indexing - Access structures which are used by applications to speed up the retrieval of records in response to certain search conditions.
- Viewing - Virtual tables in which data from underlying base tables are combined so that applications can work with just one virtual table instead of the several or more complete base tables.
- Query optimization - Heuristic and cost-based query optimization mechanisms to improve the efficiency of query execution.
- Stored Procedures - Named and precompiled set of SQL statements which are stored in the server's data dictionary and can be executed by applications through names.
- De-normalization - A process in which columns belonging to one table is redundantly defined in another table to reduce or eliminate the need to query the original table.

5.1.2.2 Reliability, Maintainability, and Availability (RMA)

The RMA of the database is a part of the RMA of the LPS. To support LPS operations 24 hours per day, 7 days per week, on a continuous basis, to provide an Operational Availability (A_O) of 0.96 or better, to achieve a mean time to restore (MTTRes) capability of 4 hours or better, and to comply with the requirement of not exceeding twice the required MTTRes in 99 percent of failure occurrences, the LPS DBMS must support the following capabilities:

- On-line backups and recovery

- On-line archiving
- Mirroring of critical database files (TBD)

5.1.2.3 Data Integrity

Data integrity is a data quality issue. Four categories of data integrity must be specified and enforced through appropriate mechanisms:

- Entity integrity - Integrity that guarantees all primary key attribute values are not null. This capability should be supported by the DBMS.
- Domain integrity - Integrity that enforces attribute values to adhere to the underlying application domain definitions. This capability should be supported by the DBMS.
- Referential integrity - Integrity that guarantees the existence and correctness of the required relationship between two entities. This capability should be supported by the DBMS.
- LPS application integrity - Integrity that protects the validity of attribute values. This capability is not provided by the DBMS and requires application software.

5.1.3 Operational Requirement Analysis

The LPS database must comply with the general operational requirements for the LPS.

5.1.3.1 Security

The LPS administrative function must provide security features to control how a database is accessed and used. Database security can be accomplished at multiple levels from applications to specific users by defining constraints that are stored in the data dictionary.

To gain access to the LPS database, each user must supply an account ID and password.

Access and use of actual LPS data objects such as tables, views, indices, and stored procedures will be controlled by grants on these

database objects. This is accomplished by defining what data objects are available to each user, user group, or application. These constraints are stored in the LPS data dictionary.

5.1.3.2 Backup and Recovery

The LPS administrative function must be able to regularly schedule backups of all or selected parts of the database while the database is operational, and be able to perform these backups on demand. Critical database tables and files must be backed up more frequently.

In the event of a failure in LPS, if there has been no destruction of the data storage devices, normal restart procedures will restore the database to a state that is ready to begin the processing.

If a data storage device was destroyed, the data on that device will be restored from a backup copy. The data will be restored to the time of the backup. Modifications of critical data files that were applied after the backup will be reapplied by the DBMS if the mirrored log files are available.

5.1.3.3 Archival and Restoral

As historical tables grow large, they can begin to affect database performance. Utilities to copy historical data onto archival storage and to delete them from the active database should be supported. Historical data will be maintained on the active database for up to a TBR period.

5.1.4 Programmatic Requirement Analysis

Each LPS subsystem interacts with the database. To permit efficient and effective programming, the LPS DBMS will:

- Be a relational database.
- Comply with ANSI/ISO SQL.
- Interface with the C language through embedded SQL.
- Provide capabilities to generate and access stored procedures.

Automated routines and user interface facilities will be implemented to support the population of LPS database.

5.1.5 High Level Entity Relationship Model

The LPS Entity Relationship (ER) model is a conceptual representation of the data for LPS application. The ER model is expressed in terms of entities in the LPS environment, the relationships or associations among those entities, and the properties of the entities and their relationships. Entities represent data items that play a functional role in the LPS application and have their own set of attributes. Table 5-1 contains a description of each LPS entity identified during the conceptual design process. The attributes of each entity are analyzed and described in section 5.2. Note that new entities may be added and existing entities may be modified, merged, or split as the project progresses and the definitions of entities become more precise.

Table 5.1 LPS Entity Descriptions (1 of 2)

Entity Name	Entity Description
Contact_Schedules	A set of contact periods containing the start and stop times when Landsat 7 spacecraft down links the wide band data to the LGS. The schedule is coming from the LGS in a hard copy form.
IDP_Acct	Aggregate accounting information for the IDPS which includes band files, browse files, and ACCA account information on a subinterval basis.
LDT_Output_File_ Info	All state information about LPS output files that are of concern to LDTS.
LPS_Configuration	The set of parameters used to configure the LPS. Some parameters are used when the system starts up while others are used during the processing.
MFP_Acct	An aggregate accounting information from the MFPS which includes the Level 0R quality and accounting information on a subinterval basis.
PCD_Acct	An aggregate accounting information from the PCDS which includes the processed PCD quality and accounting information on a subinterval basis.

Table 5.1 LPS Entity Descriptions (2 of 2)

Entity Name	Entity Description
RDC_Acct	Raw data capture accounting information on a contact basis.
RDP_Acct	An aggregate accounting information from the RDPS which includes return link quality and accounting information on a contact basis.
Sub_Intv	A list of subinterval information generated by the MFPS and used by the PCDS, IDPS, and MACS to generate LPS files.
Valid_Band_Parms	An aggregate information which includes band parameters and reduction ratio for browse and ACCA processing.
Valid_CCSDS_Parms	A list of parameters that controls CCSDS frame synchronization and bit slip correction.
Valid_MFP_Parms	The validated MFPS setup parameters.
Valid_MFP_Thres	The validated MFPS threshold values.
Valid_PCD_Parms	The validated PCD parameters used in processing PCD data.
Valid_PCD_Thres	The validated PCD threshold parameters used in processing PCD data.
Valid_RDP_Thres	Validated RDP processing thresholds.
Valid_Scene_Parms	The validated general mission information and parameters provided by IAS and used to calculate the longitude, latitude, the WRS scene Id and sun elevation and azimuth.
Valid_WRS_Parms	The validated Worldwide Reference System table containing the information for each WRS scene.

Figure 5-2
LPS ER Diagram

The LPS ER model is expressed as ERDs, which are graphical representations of the ER model as illustrated in Figure 5.2. In addition to entities described above, relationships are also included in the diagram. The relationships represent the association between the instances of one or more entities that are of interest to the LPS. For each relationship, there is a cardinality associated with it. The cardinality describes the number of instances of one entity that can be associated with each instance of the entity to which it relates. For instance, there is more than one scene information (Scene_Parms) for each subinterval (Sub_Intv) while there is only one MFPS accounting information (MFP_Acct) for each subinterval.

5.2 Logical Design

The logical design process is concerned with transforming the ERDs from the conceptual design phase to a logical relational model. Four major activities were performed during the logical design phase.

1. Represent entities. Each entity in the ERDs is represented as a logical schema in the relational data model. Candidate and primary keys are determined. The candidate keys are the sets of attributes that uniquely identify one occurrence of an entity. The predominant set of key attributes is the primary key.

2. Represent relationships. Each relationship in the ERDs is represented in the relational data model either by making the primary key of one logical schema a foreign key of another logical schema or by creating a separate logical schema. A foreign key is an attribute that appears as a nonkey attribute in one schema and as a primary key attribute in another schema.

3. Normalize the schemata. The logical schemata that are created in step 1 and 2 may have unnecessary redundancy and may be subject to anomalies when they are updated. Normalization is a process that refines the logical schemata to avoid these problems.

4. Merge the schemata. In some cases schemata may be redundant that must be merged to remove the redundancy.

The result of logical design is used as the foundation for the physical database design.

5.2.1 Logical Schema Definitions

To create well structured schemata that contain a minimum amount of redundancy and allow users to insert, modify, and delete database information without errors or inconsistencies, normalization principles are applied during the logical design process. The steps in normalization are illustrated in Figure 5.3.

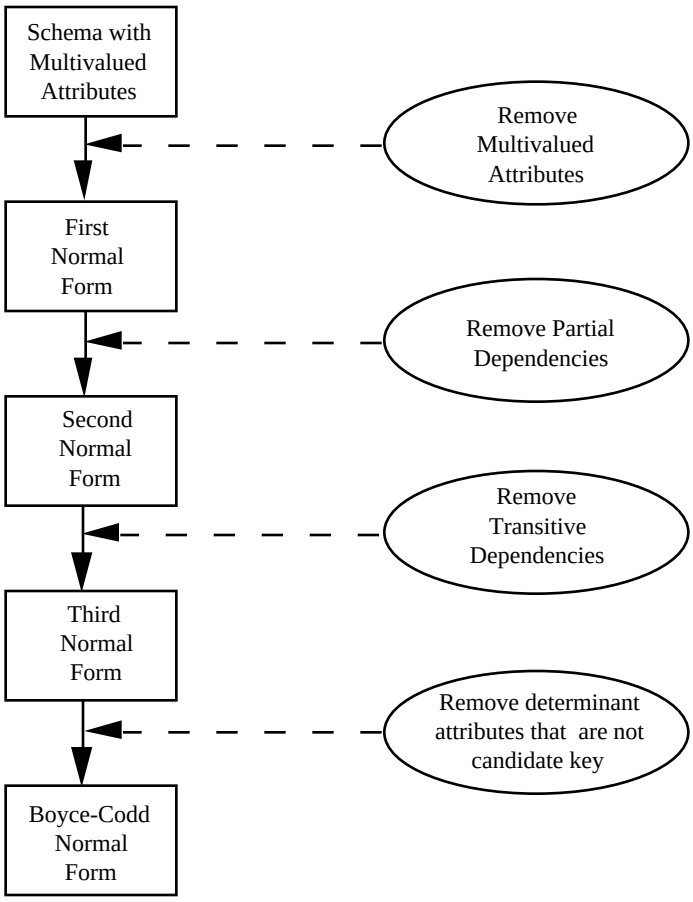


Figure 5.3 Normalization Steps

Most of the LPS logical schemata are normalized to the standard Boyce-Codd normal form. Some of the schemata are not normalized due to the fact that their attributes are yet TBD or TBR. Normalization and merging of schemata will continue until all schemata are well structured. The LPS logical schemata as of this phase are listed in Table 5.2. The primary key columns are identified by asterisks (*). Note that the attributes and primary keys of some threshold schemata are not yet determined due to incompleteness of schema definitions.

Table 5.2 LPS Logical Schema Definitions (1 of 8)

Schema Name	Attribute Name	Attribute Description
Contact_Schedules	*Contact_Schedule_Id	Contact schedule identification
	Contact_Schedule_Start_Time	Contact schedule start time
	Contact_Schedule_Stop_Time	Contact schedule stop time 0
IDP_Acct	*Sub_Intv_Sequence_Id	Subinterval sequence identification - A surrogate key for subinterval
	*File_Name	Band file or browse file name
	Bands_Present	Bands present (multi -valued)
	File_Type	Band, mono or multiband browse
	CCA_Method	CCA method
LDT_Output_File_Info	*LPS_Hardware_String_Id	LPS hardware string id
	*LGS_Channel_Id	Contact channel id
	*Contact_Start_Time	Contact start time
	*Contact_Stop_Time	Contact end time
	DAN_Suspended	Indicator of DAN transfer suspension
	Marked_For_Retention	Flag indicating deletion of files by LPS
	Time_Available	Output file for transfer available time
	Time_Deleted	Output file deletion time
	DAN_Transmission_Time	DAN transmission time from LPS to DAAC
	DTA_Time_Of_Receipt	DTA from LP DAAC receipt time by LPS
	Contact_File_Names	File names associated with each subinterval (multi -valued)
LPS_Configuration	*LPS_Hardware_String_Id	LPS hardware string ID
	LGS_Channel_Id	LGS channel identification
	Spacecraft_Id	Landsat 7 spacecraft ID
	Instrument_Id	Instrument ID (ETM+)
	LPS_Software_Version_Number	LPS software version number

Table 5.2 LPS Logical Schema Definitions (2 of 8)

Schema Name	Attribute Name	Attribute Description
	File_Version_Number	Metadata file version no of a subinterval
MFP_Acct	*Sub_Intv_Sequence_Id #	Subinterval sequence identification - A surrogate key for subinterval
	Mjf_CADU_Rcvd_Cnt	Count of received major frame CADUs
	Mjf_CADU_Fly_Cnt	Count of flywheel major frame CADUs
	Mjf_CADU_Polarity	CADU sync info polarity of accumul. mjf (multi -valued)
	CADU_Search_Tolerance	Search tolerance parameters
	CADU_Check_Tolerance	Check tolerance parameter
	CADU_Flywheel_Tolerance	Flywheel tolerance parameter
	CADU_Sync_Marker_ Check_Error_Tolerance	Check error tolerance parameter
	CADU_Sync_Lock_Error_ Tolerance	Lock error tolerance parameter
	CADU_Bit_Slip_ Correction_Extent	Slip correction extent parameter
	Mjf_CADU_Bit_Slip	Bit slip total for accum. major frame set (multi -valued)
	Mjf_CADU_Sync_Err_Cnt	Count of major frame CADUs with synchronization errors
	Mjf_CADU_Missing_Cnt	Number of missing CADUs per major frame
	Mjf_CADU_RS_Corr_Cnt	Number of correctable VCDU headers per major frame
	Mjf_CADU_RS_Uncorr_Cnt	Number of uncorrectable VCDU header per mjf
	Mjf_CADU_BCH_Corr_Cnt	No of CADUs with BCH error corrected in the mission data zone per major frame

	Mjf_CADU_BCH_Uncorr_Cnt	No of CADUs with BCH error uncorrected in the mission data zone per major frame
	Mjf_CADU_BCH_Bits_Corr	Total no of bits of BCH corrected in the mission data zone of a CADU

Table 5.2 LPS Logical Schema Definitions (3 of 8)

Schema Name	Attribute Name	Attribute Description
	Mjf_CADU_CRC_Err_Cnt	Count of CADUs with CRC errors
	Mjf_CADU_BER_Cnt	BER in mission data zone on a subint basis
	Mjf_CADU_Seq_Err_Cnt	No of VCDU counter errors in a maj. frame
	ETM_Data_Format	ETM + data format type provided by MFPS and is included in the metadata file
	Mjf_Count	Count of major frames in subinterval
	Mjf_Tossed_Cnt	No of mjf calculated from sync errors and end of line code errors on a subint basis
	Mjf_Eol_Err_Cnt	No of end of line errors on a subint basis
	Mnf_Ctr_Err	No of minor frame counter errors
	Mjf_Time_Code_Err_Cnt	Count of imagery time code errors
	Mjf_Full_Fill_Cnt	Count of entirely filled ETM+ major frames
	Mjf_Part_Fill_Cnt	Count of partially filled ETM+ major frames
	Cal_File_Name	Calibration data file name
	MSCD_File_Name	Mirror Scan Correction Data file name
PCD_Acct	*Sub_Intv_Sequence_Id	Subinterval sequence identification - A surrogate key for subinterval
	Orbit_Number	Orbit number
	Num_PCD_MJF	Number of PCD major frames
	First_PCD_MJF_Time	First PCD major frame time
	PCD_File_Id	PCD file name
	Failed_PCD_Votes	Number of failed PCD votes

	Num_PCD_MNF_Sync_ Errors	Number of PCD minor frames with sync errors
	Num_PCD_Filled_MNF	Number of PCD filled minor frames
	Num_PCD_Filled_MJF	Number of PCD filled major frames
	Num_Avail_ADP	Number of data points
	Num_Rejected_ADP	Number of data points rejected
	Num_Missing_ADP	Number of data point missing
	Num_Avail_EDP	Number of data points

Table 5.2 LPS Logical Schema Definitions (4 of 8)

Schema Name	Attribute Name	Attribute Description
	Num_Rejected_EDP	Number of data points rejected
	Num_Missing_EDP	Number of data point missing
RDC_Acct	*LPS_Hardware_String_Id	LPS hardware string id
	*LGS_Channel_Id	Contact channel id
	*Contact_Start_Time	Contact Start Time
	*Contact_Stop_Time	Contact end time
	Rcv_Dat_Vol_Mbytes	Received data volume
RDP_Acct	LPS_Hardware_String_Id	LPS hardware string id
	LGS_Channel_Id	Contact channel id
	Contact_Start_Time	Contact start time
	Contact_Stop_Time	Contact end time
	CADU_Search_Tolerance	Search tolerance parameter
	CADU_Check_Tolerance	Check tolerance parameter
	CADU_Flywheel_ Tolerance	Flywheel tolerance parameter
	CADU_Sync_Marker_ Check_Error_Tolerance	Check error tolerance parameter
	CADU_Sync_Lock_Error_ Tolerance	Lock error tolerance parameter
	CADU_Bit_Slip_ Correction_Extent	Slip correction extent parameter
	CADU_Polarity	CADU polarity (multi -valued)
	CADU_Bit_Slip	Bit slip total for CADU (multi -valued)
	CADU_Sync_Error_Count	Count of CADUs with synchronization errors
	CADU_Rcv_Count	Count of received CADUs
	CADU_Flywheel_Count	Count of flywheel CADUs
	CADU_Missing_Count	Count of missing CADUs
	CADU_CRC_Error_Count	CRC errors encountered when CCSDS was processing a raw wideband data set

Table 5.2 LPS Logical Schema Definitions (5 of 8)

Schema Name	Attribute Name	Attribute Description
	VCDU_Header_ Correctable_Error_Count	Count of correctable VCDU headers, by VCDU-ID (Reed Solomon checked)
	VCDU_Header_Uncor-rectable_Error_Count	Count of uncorrectable VCDU headers (Reed Solomon checked)
	BCH_Data_Corrected_ Error_Count	Count of CADUs with BCH errors corrected for the mission data zone in the VCDU
	BCH_Data_Uncorrected_ Error_Count	Count of CADUs with BCH errors uncorrected for the mission data zone in the VCDU
	BCH_Pointer_Corrected_ Error_Count	No of correctable BCH pointer field errors encountered when CCSDS was processing a raw wideband data set
	BCH_Pointer_Uncorrected _ Error_Count	No of uncorrectable BCH pointer field errors encountered when CCSDS was processing a raw wideband data set
	BER	Bit error rate
Sub_Intv	*LPS_Hardware_String_Id	LPS hardware string id
	*LGS_Channel_Id	Contact channel id
	*Contact_Start_Time	Contact start time
	*Contact_Stop_Time	Contact end time
	*Sub_Intv_Sequence_Id	Subinterval sequence identification - A surrogate key for subinterval
	MF_Start_Time	Subinterval start time
	MF_Stop_Time	Subinterval stop time
	VCID	Virtual channel identification
Valid_Band_Parms	Mono	Monochrome browse band
	Multi1	Multiband browse band
	Multi2	Multiband browse band
	Multi3	Multiband browse band
	Subs	Subsampling reduction ratios

Table 5.2 LPS Logical Schema Definitions (6 of 8)

Schema Name	Attribute Name	Attribute Description
	Wave	Wavelet reduction ratios
	CCA_Method	CCA method
	CCA_Ratio	CCA ratio
Valid_CCSDS_Parms	CADU_Search_Tolerance	Search tolerance parameter
	CADU_Check_Tolerance	Check tolerance parameter
	CADU_Flywheel_ Tolerance	Flywheel tolerance parameter
	CADU_Sync_Marker_ Check_Error_Tolerance	Check error tolerance parameter
	CADU_Sync_Lock_Error_ Tolerance	Lock error tolerance parameter
	CADU_Bit_Slip_ Correction_Extent	Slip correction extent parameter
Valid_MFP_Parms	Sensor_Alignment_Info	Information to perform integer-pixel alignment (multi-valued)
	Fill_Value	Fill value for major frame processing
	Sub_Intv_Delta	Delta for determining subinterval
	Mjf_Data_Rate	Data rate threshold
	Max_Alignment_Value	Maximum alignment value
	Time_Range_Tol	Time range tolerance
	Part_Mnf_Tol	Minor frame tolerance
	Maj_Vote_Tol	Majority voting tolerance
Valid_MFP_Thres	Mjf_CADU_Seq_Err_Thr	Threshold value for the number of sequence counter errors
	Scan_Dir_Thr	The counter thresholds for the VCDU identification process
	Sync_Thr	The maximum number of sync error allowed
	Mnf_Ctr_Thr	The maximum number of minor frame errors allowed
	Eol_Thr	The maximum number of end of line error allowed

Table 5.2 LPS Logical Schema Definitions (7 of 8)

Schema Name	Attribute Name	Attribute Description
	Tc_Thr	The maximum number of time code error allowed
	Full_Mjf_Thr	The maximum number of full filled major frame allowed
	Part_Mjf_Thr	The maximum number of partial filled major frame allowed
Valid_PCD_Parms	PCD_Frame_Fill_Value	Predefined value that is used to fill missing PCD data when building PCD minor and major frames
Valid_PCD_Thres	Ephem_Position_Upper	Largest valid ephemeris position data pnt
	Ephem_Position_Lower	Smallest valid ephemeris position data pnt
	Ephem_Velocity_Upper	Largest valid ephemeris data point
	Ephem_Velocity_Lower	Smallest valid ephemeris data point
	Att_Lower_Bounds	Lowest valid value of any attitude component
	Att_Upper_Bounds	Highest valid value of any attitude component
	Num_Missing_Data_ Words	The threshold for reporting errors when PCD information words are missing
	Num_Failed_Votes	The threshold for reporting errors when unsuccessful majority votes are performed
Valid_RDP_Thres	Sync_Thres	Max no of sync errors allowed before notifying the operator
	CRC_Thres	Max no of CRC errors allowed before notifying the operator
	RS_Thres	Max no of Reed Solomon errors allowed before notifying the operator
	BCH_Thres	Max no of BCH errors allowed before notifying the operator

	BER_Thres	Max no of bit error rate allowed before notifying the operator

Table 5.2 LPS Logical Schema Definitions (8 of 8)

Schema Name	Attribute Name	Attribute Description
Valid_Scene_Parms	ETM_Plus_To_Body_Trans	Parms used to transform the latitude, longitude, sun elevation, and sun azimuth from ETM Plus to Spacecraft Body
	Mission_Start_Time	Start time of the Landsat Mission
	Time_Per_Orbit	Amount time required for Landsat to make one complete orbit
	Semi_Major_Axis	Distance from Apogee or Perigee to the center of the orbit ellipse
	Semi_Minor_Axis	Polar axis radius
	ETM_Plus_LOS_x	X-coordinate of the line of sight vector
	ETM_Plus_LOS_y	y-coordinate of the line of sight vector
	ETM_Plus_LOS_z	Z-coordinate of the line of sight vector
Valid_WRS_Parms	*WRS_Path_Nominal	WRS path number
	*WRS_Row_Nominal	WRS row number
	Scene_Center_Latitude	Scene center latitude
	Scene_Center_Longitude	Scene center longitude
	Upper_Left_Corner_ Latitude	Upper left corner latitude
	Upper_Right_Corner_ Latitude	Upper right corner latitude
	Lower_Left_Corner_ Latitude	Lower left corner latitude
	Lower_Right_Corner_ Latitude	Lower right corner latitude
	Upper_Left_Corner_ Longitude	Upper left corner longitude
	Upper_Right_Corner_ Longitude	Upper right corner longitude
	Lower_Left_Corner_ Longitude	Lower left corner longitude

	Lower_Right_Corner_ Longitude	Lower right corner longitude

5.2.2 Functional Usage Analysis

A preliminary data usage analysis is performed during the logical design process which examines the manner in which the LPS subsystems and processes interact with the database. The functional usage of the database can be presented using interface diagrams and CRUD matrices. Table 5.3 is a CRUD matrix that depicts the interactions between LPS subsystems and database schemata. Convention: I - Insert, U - Update, D - Delete, Q - Query.

Table 5.3 LPS Subsystem CRUD Matrix

Schema\Subsystem	RDCS	RDPS	MFPS	PCDS	IDPS	MACS	LDTS
Contact_Schedules						I,U,Q	
IDP_Acct					I,U,Q	Q	
LDT_Output_File_Info							I,U,Q
LPS_Configuration	Q		Q	Q	Q	I,U,Q	
MFP_Acct			I,U,Q			Q	
PCD_Acct				I,U,Q		Q	
RDC_Acct	I,U,Q						
RDP_Acct		I,U,Q	Q				
Sub_Intv			I,U,Q	Q	Q	Q	
Valid_Band_Parms					I,U,Q		
Valid_CCSDS_Parms		I,U,Q					
Valid_MFP_Parms			I,U,Q				
Valid_MFP_Thres			I,U,Q				
Valid_PCD_Parms				I,U,Q			
Valid_PCD_Thres				I,U,Q			
Valid_RDP_Thres		I,U,Q					
Valid_Scene_Parms				I,U,Q			
Valid_WRS_Parms				I,U,Q		Q	

Figure 5.4 through 5.10 illustrate the interface between major processes in each subsystem and the database. The CRUD matrix for each subsystem is also included in the following figures.

Schema\Process	Receive Raw Wideband Data	Restage Raw Wideband Data	Generate Data Receive Summary
LPS_Configuration	Q		
RDC_Acct	I,U	I,U	Q

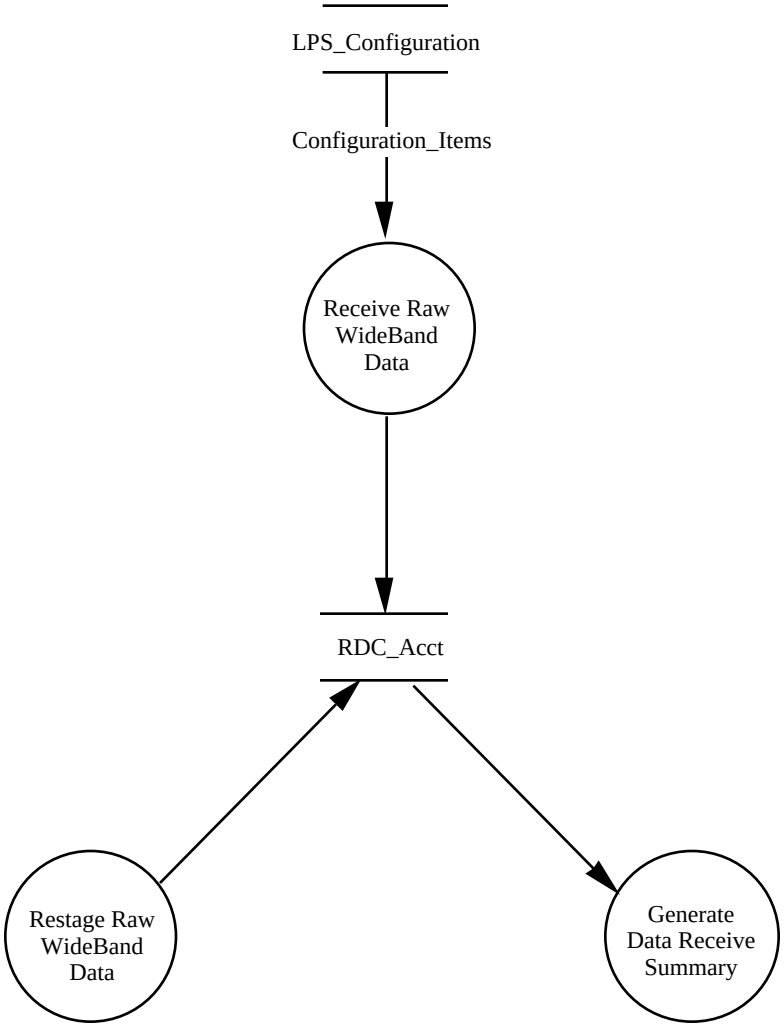


Figure 5.4 RDCS Database Interface

Schema\Process	Validate RDP Para-meters	Synchro-nize CCSDS Frame	Process CCSDS Grade 3	Decode BCH	Compute BER	Generate Return Link QA Report
RDP_Acct		I,U,Q	I,U,Q	I,U,Q	Q	Q
Valid_CCSDS_Parms	I,U	Q				
Valid_RDP_Thres	I,U	Q	Q	Q		Q

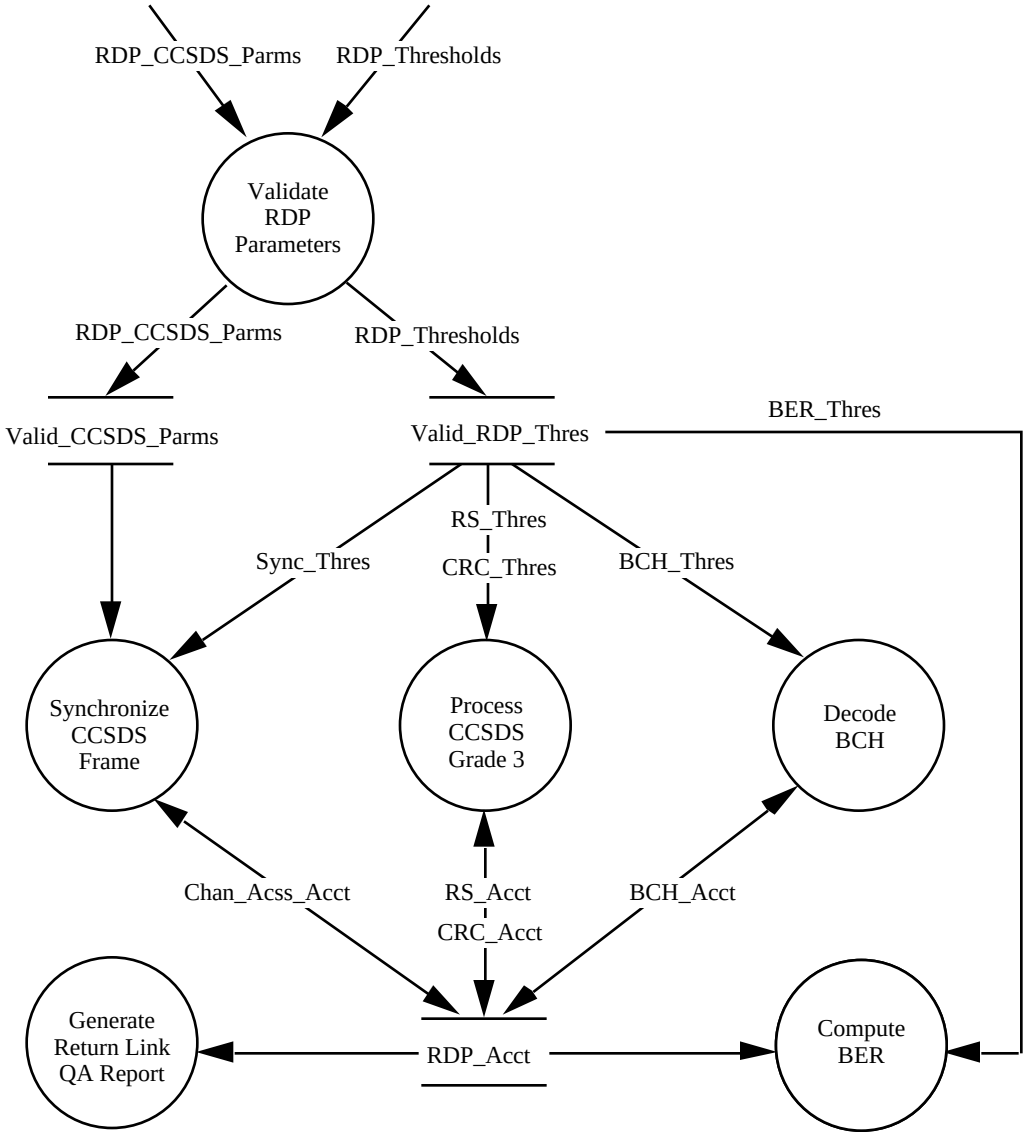


Figure 5.5 RDPS Database Interface

Schema\ Process	Vali-date MFP Para-mets	Coll-ect VCDU Qty & Acct	Iden-tify VCDU	Parse Major Frme	Col-lect Qlty & Acct	Gene-rate Band Data	Ext-ract Calib Data	CreateC alib File	Create MSCD File	Gene-rate Repor t
LPS_Con-figuration								Q	Q	
MFP_Acct					I,U			I,U	I,U	Q
RDP_Acct		Q		Q						
Sub_Intv				I,U				Q	Q	Q
Valid_MFP _ Parms	I,U			Q		Q	Q			
Valid_MFP _ Thres	I,U		Q		Q					

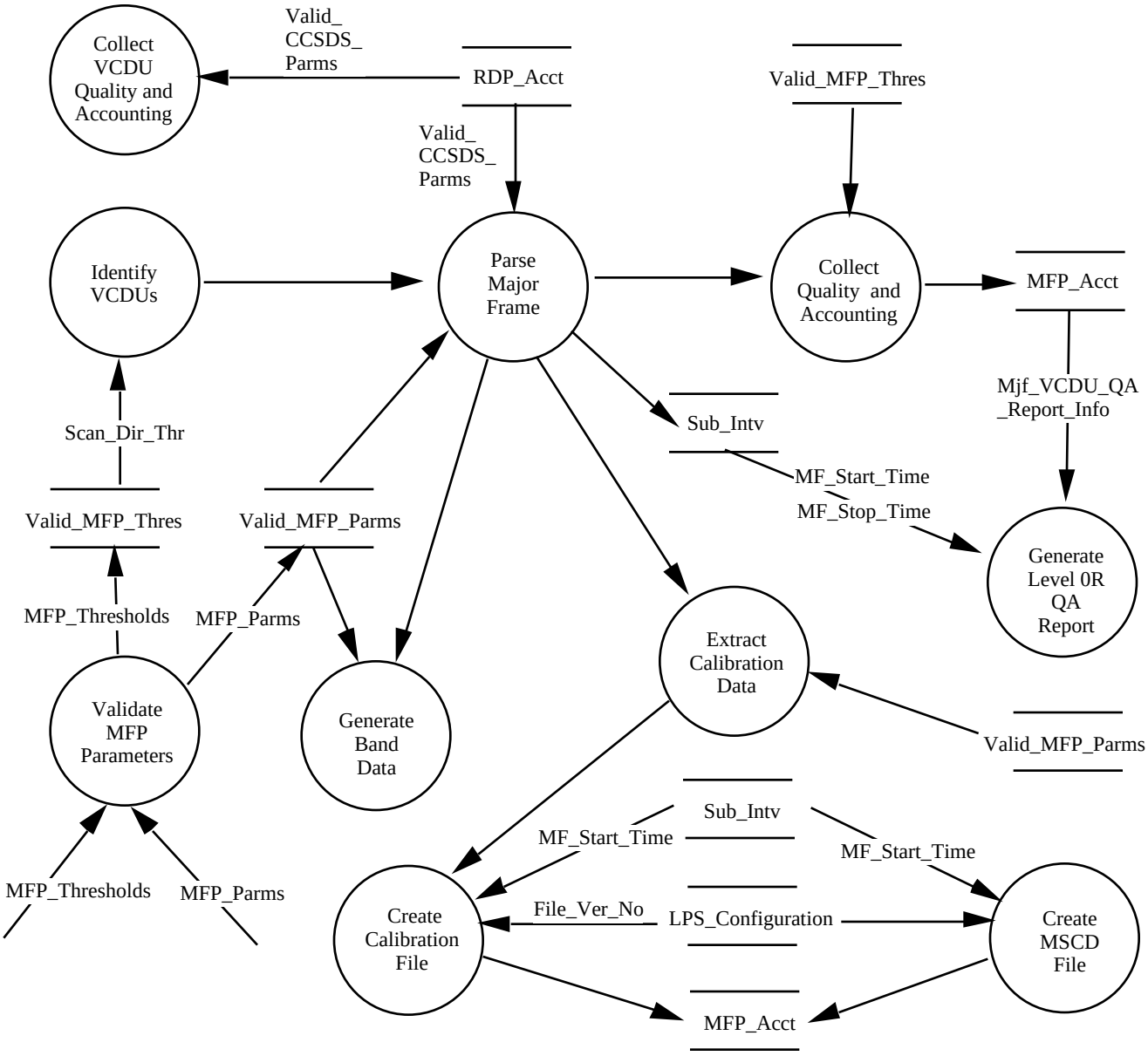


Figure 5.6 MFPS Data Interface

Schema\Process	Validate PCD Parameters	Assemble PCD Cycles	Calculate Scene Info	Create PCD File
LPS_Configuration				Q
PCD_Acct		I,U	I,U	I,U
Sub_Intv			Q	Q
Valid_PCD_Parms	I,U	Q		
Valid_PCD_Thres	I,U	Q		
Valid_Scene_Parms	I,U		Q	Q
Valid_WRS_Parms	I,U		Q	

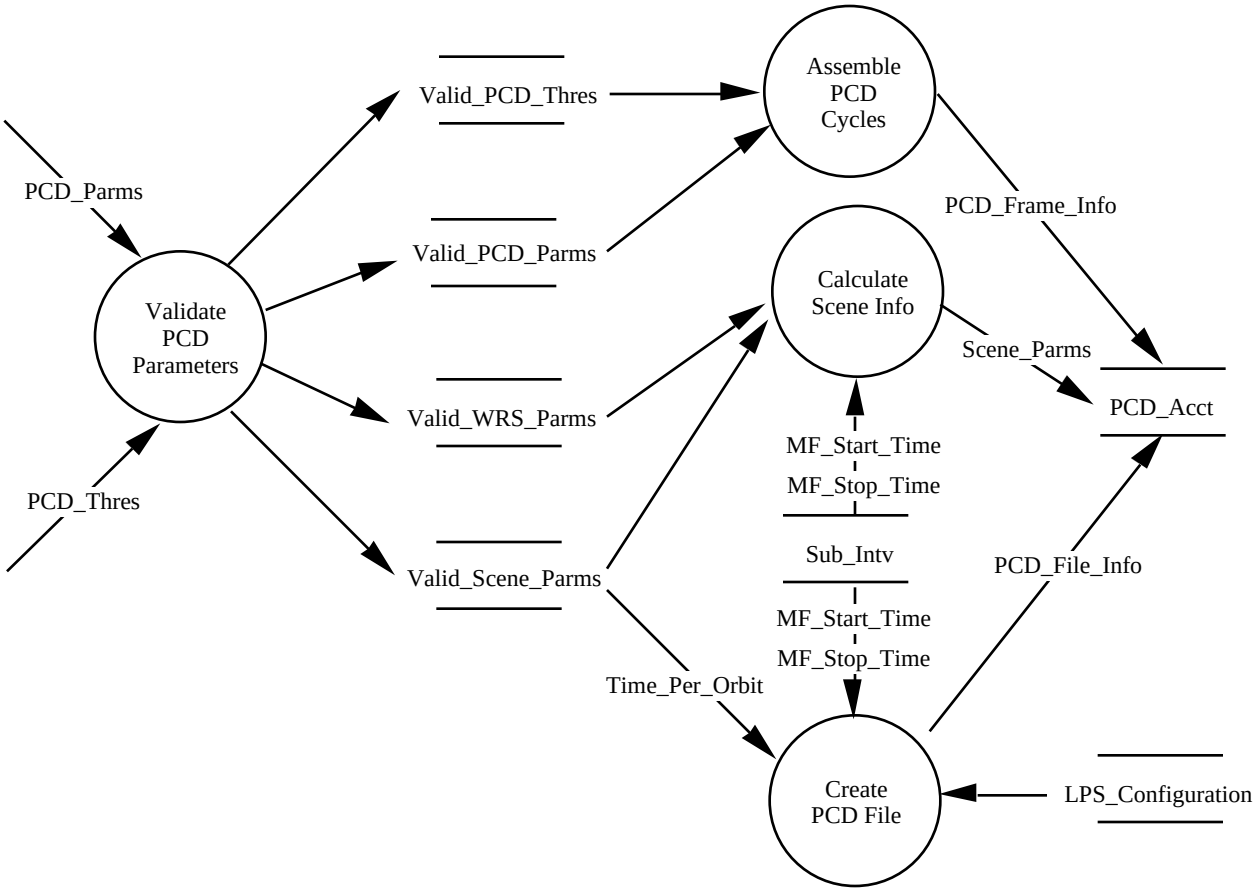


Figure 5.7 PCDS Database Interface

Schema\Process	Validate Band Parameter	Generate Browse File	Generate Band File	Perform ACCA
IDP_Acct		I,U	I,U	I,U
LPS_Configuration		Q	Q	
Sub_Intv		Q	Q	
Valid_Band_Parms	I,U	Q		Q

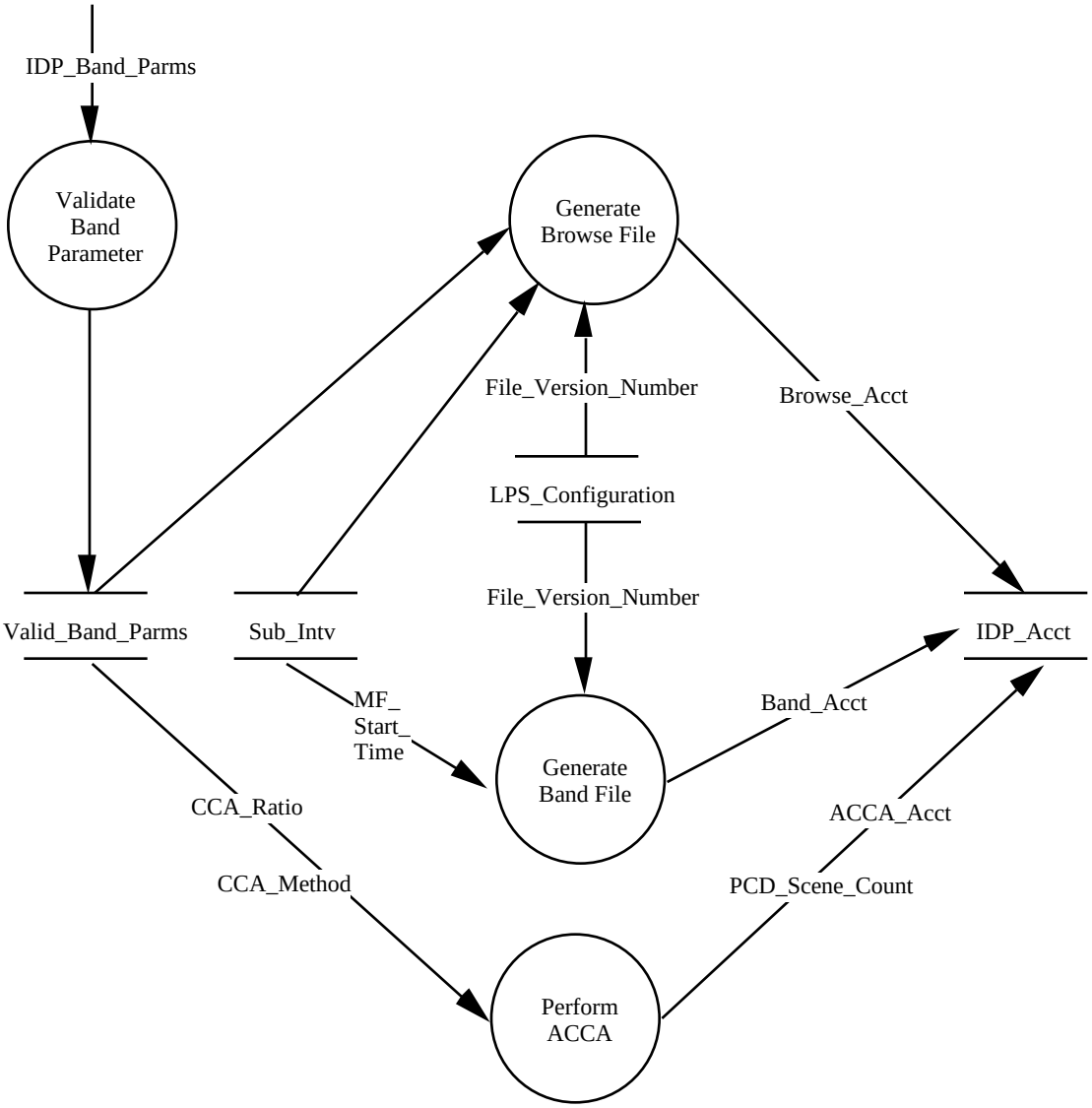


Figure 5.8 IDPS Database Interface

Schema\Process	Modify LPS Configuration	Modify Contact Schedule	Generate Metadata
Contact_Schedules		I,U,Q	
IDP_Acct			Q
LPS_Configuration	I,U,Q		Q
MFP_Acct			Q
PCD_Acct			Q
Sub_Intv			Q
Valid_WRS_Parms			Q

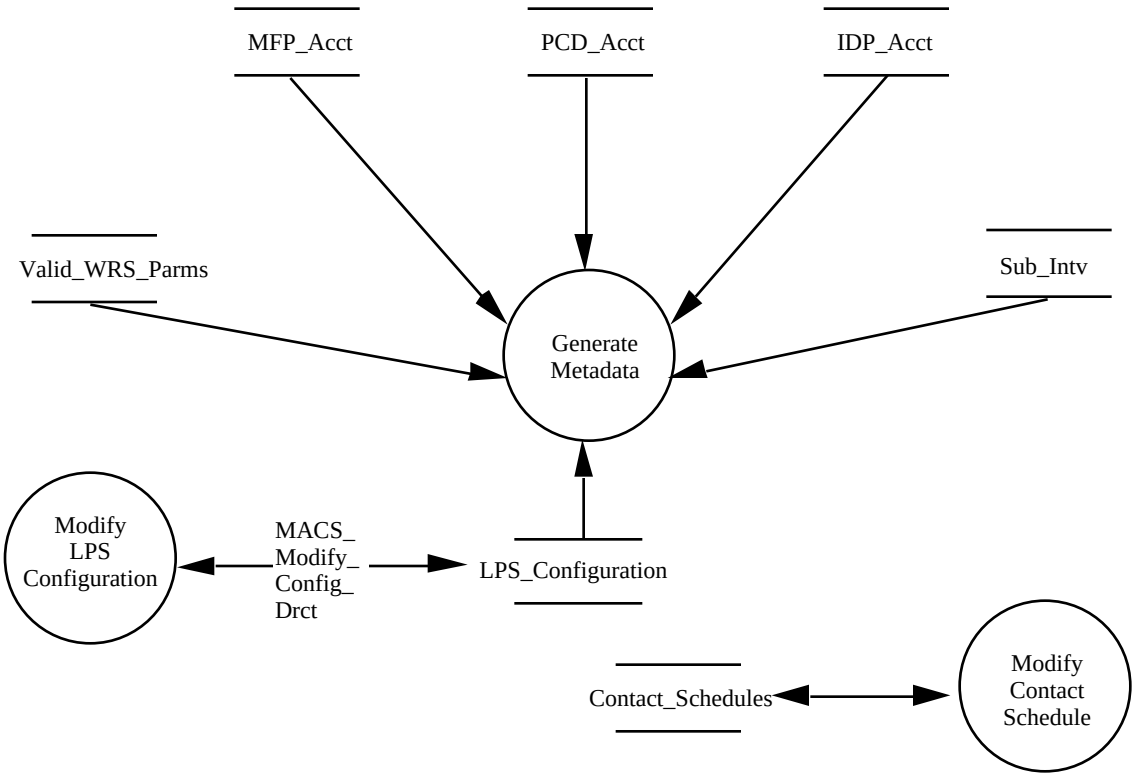


Figure 5.9 MACS Database Interface

Schema\ Process	Generate DAN	Send DAN	Control Send DAN	Delete LPS Files	Retain LPS Files	Receive DTA	Generate Transfer Summary
LDT_Output _File_Info	I,U	I,U,Q	Q	I,U,Q	I,U	I,U	Q

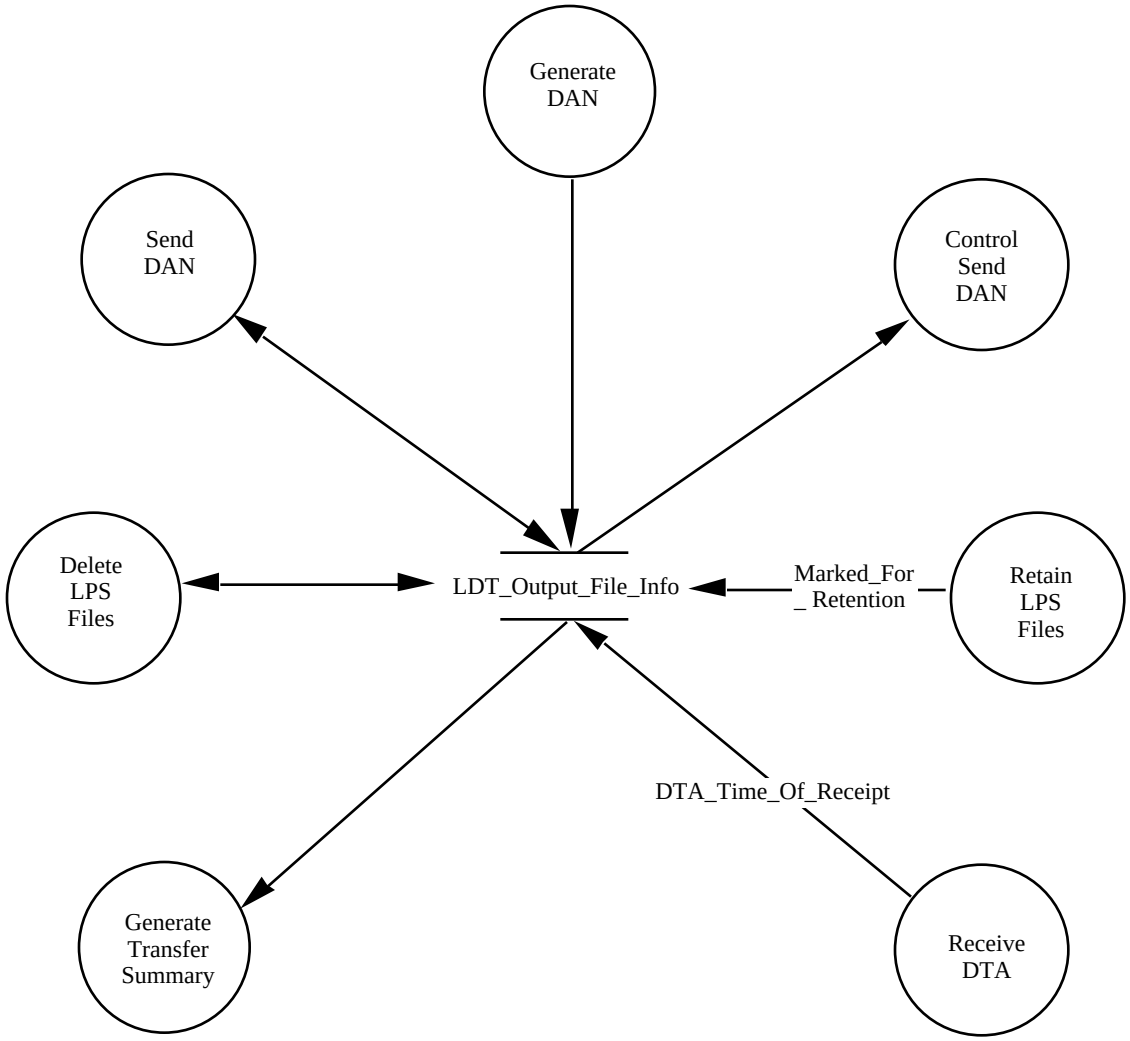


Figure 5.10 LDTS Database Interface

6.0 User Interface (UI)

The LPS user interface consists of all elements of interaction between LPS and the operator. For LPS, this interaction will probably be some combination of system level commands and Oracle user interface products. There is the possibility that existing products, such as a process manager ('DPCP' described in section 6.3.2 Reusability), will be used.

Concurrent development of the software requirements and the user interface offers specific advantages. By examining the interactions expected between the operators and the system, we can avoid potential problems and aid the understanding of what the subsystems must do in order to perform their function. Another important benefit to beginning work on the user interface is that software drivers may be uncovered by close inspection of how the system will be required to operate. Finally, early effort can lead to a preliminary user interface, offering the chance for users to provide input into the design decisions of both the final user interface and the application software.

Preliminary analysis in the user interface area has covered the areas of task analysis, performance goal setting, and user interface mock-up. The results of these studies are presented in the next three sections.

6.1 Task Analysis

The task analysis provides a conceptual framework from which to approach the design of the user interface. It explains drivers and limitations on what the user interface will or will not be.

6.1.1 Drivers

- The operator is required to setup, test, monitor, and control the LPS system.
- Each LPS string is physically and logically independent. Each string must have its own user interface.
- Operations can be performed on several contacts at one time. The processing of contact 1 can still be going on while contact 2 needs to be captured.

6.1.2 Constraints

w The budget for the user interface is very limited. Many choices will be based on budgetary constraints.

6.1.3 Assumptions

w No network interface will be available for input of schedules, parameters, etc. between LPS and external systems.

w No security will be provided other than what is available from the UNIX shell and from ORACLE.

w There will be only one type of user for the LPS system. This user type is classified as an "operator".

w The LPS operators will be capable of utilizing the operating system to perform some of the user interface functions. No elaborate shell program is needed to buffer the operator from UNIX.

w No long term or trend reporting is required.

6.1.4 Decisions

w The user interface will be developed as some combination of UNIX shell commands, ORACLE SQL-MENUS, and ORACLE SQL-FORMS. It is possible that some COTS or reusable code may be identified and used.

w There will be no automated coordination between strings. This implies that there will be no system wide reporting.

w Prototype user interface screens will be developed and reviewed by LPS operators.

6.1.5 User Interface Event List

This section contains a complete list of all known interactions between the LPS system and the operator.

SYSTEM CONFIGURATION

F&PS	EVENT	SUBSYSTEM	Data Dictionary
3.1.14	Configure LPS (normal/fall back)	System Level	None

THRESHOLDS

F&PS	EVENT	SUBSYSTEM	Data Dictionary
3.3.6.6	Input RDPS Thresholds	RDPS	RDP_Thresholds
3.3.6.6	Input MFPS Thresholds	MFPS	MFP_Thresholds
3.3.6.6	Input PCDS Thresholds	PCDS	PCD_Thresholds

PARAMETERS

F&PS	EVENT	SUBSYSTEM	Data Dictionary
3.3.6.1	Input LPS configuration	MACS	LPS_Configuration
3.2.1	Input contact schedules (from LGS)	MACS	Contact_Schedules
derived	Input MFP Parameters	MFPS	MFP_Parms
3.3.2.22	Input sensor align.tables (from IAS)	MFPS	Sensor_Alignment_Info
3.3.2.1	Input CCSDS AOS grade 3 parameters	RDPS	RDP_CCSDS_Parms
3.3.3.2	Input Browse Monochrome Band	IDPS	IDP_Band_Parms
3.3.3.3	Input Browse Multi Band 1,2&3	IDPS	IDP_Band_Parms
3.3.4.10	Input ACCA comparison values	IDPS	IDP_Band_Parms
derived	Input PCD Parameters	PCDS	PCD_Parms

TEST

F&PS	EVENT	SUBSYSTEM	Data Dictionary
3.1.10.6	Test functions and external interfaces	System Level	None
3.1.10.7	Execute diagnostic tests	System Level	None
3.1.10.8	Support end-to-end testing of LPS functions	System Level	None
3.1.19	Read test points to verify proper operation	System Level	None

CONTROL

F&PS	EVENT	SUBSYSTEM	Data Dictionary
3.1.10.1	Startup LPS	IRIX	None
3.1.10.2	Shutdown LPS	IRIX	None
3.1.11	Control LPS operations		
3.3.6.9a	- Start capture of wideband data	RDCS	RDC_Capture_Drct
3.3.6.9a	- Stop capture of wideband data	RDCS	RDC_Capture_Drct
3.3.1.7	- Start copy from disk to tape	RDCS	RDC_Save_Drct
derived	- Stop copy from disk to tape	RDCS	RDC_Save_Drct
3.3.1.9	- Start copy from tape to disk	RDCS	RDC_Restage_Drct
derived	- Stop copy from tape to disk	RDCS	RDC_Restage_Drct
3.3.6.9b	- Start Level 0R Processing	RDPS	RDP_Process_Drct
3.3.6.9b	- Stop Level 0R Processing	IRIX kill	None
3.3.6.8	Manually override automated functions	IRIX kill	None

MONITORING

F&PS	EVENT	SUBSYSTEM	Data Dictionary
3.1.12	Monitor LPS operations	ALL (MACS)	Assorted error messages
3.3.6.7	Report error threshold exceeded	All	Assorted error messages
3.3.6.7	Report result threshold exceeded	All	Assorted error messages
3.1.10.3	Report error messages	ALL (MACS)	Assorted error messages
3.1.10.4	Isolate system faults	MACS/IRIX	LPS_Journal
3.1.10.5	Recover from system faults	MACS/IRIX	LPS_Journal
derived	Examine LPS_Journal	text editor	LPS_Status

FILE MANAGEMENT

F&PS	EVENT	SUBSYSTEM	Data Dictionary
3.3.6.9c	Enable File Transfer	MACS/LDTS	LDT_Enable_File_Xfer_Drct
3.3.6.9c	Disable File Transfer	MACS/LDTS	LDT_Disable_File_Xfer_Drct
derived	Delete Raw Data Input File	RDCS	RDC_Delete_Drct
3.3.5.5	Delete output files on contact basis	LDTS	LDT_Delete_Files_Drct
3.3.5.6	Retain output files on contact basis	LDTS	LDT_Retain_Files_Drct
3.3.2.8	Examine CADU CCSDS trouble files	IRIX	n/a
3.3.2.10	Examine CADU BCH trouble files	text editor	n/a
derived	Resend DAN to LP DAAC	LDTS/MACS	LDT_Resend_DAN_Drct

REPORTS

F&PS	EVENT	SUBSYSTEM	Data Dictionary
3.3.1.10.1	Display wideband data receive summary	RDCS	RDC_Rpt_Data_Capture_Sum_Drct
3.3.1.10.1	Print wideband data receive summary	RDCS	RDC_Rpt_Data_Capture_Sum_Drct
3.3.6.4	Display Return link Q&A data (contact)	RDPS	RDP_Rpt_Return_Link_QA_Drct
3.3.6.4.1	Print Return link Q&A data (contact)	RDPS	RDP_Rpt_Return_Link_QA_Dr
3.3.6.4	Display level 0R Q&A data (subinterval)	MFPS	MFP_Rpt_L0R_QA_Drct
3.3.6.4.1	Print level 0R Q&A data (subinterval)	MFPS	MFP_Rpt_L0R_QA_Drct
3.3.6.5	Display transfer summary (contact)	LDTS	LDT_Rpt_File_Xfer_Sum_Drct
3.3.6.5.1	Print transfer summary (contact)	LDTS	LDT_Rpt_File_Xfer_Sum_Drct

6.2 User Interface Goals

This section is commonly used to define specific quantitative goals defining minimal acceptable user interface performance. Due to the limited budget of the LPS, it is undesirable to place hard restrictions on items like response time, when a slightly relaxed requirement could produce a user interface at a much lower cost. LPS will seek to develop a user interface which is responsive, easy to use, and maximizes efficient operations.

6.3 User Interface Mock-up

The user interface consists of three different types of commands - Operating System, reusable COTS software, and ORACLE SQL-MENU and SQL-FORMS.

6.3.1 Operating System

UNIX commands will be used to start the LPS system on each string.

The UNIX "tar" command can satisfy the need to start and stop the copy to short term storage.

The operating system will be used to set the priority of LPS processes.

6.3.2 Reusability

If applicable and practical, pre-existing software will be used for the LPS user interface. These reuse sources have already been described in section 3.

One possible source of such software is NASA's Ground Operations Technology Testbed (Code 520).

- The first possible reuse item identified is the Distributed Process Control Program (DPCP). This tool enables an operator to start and monitor a set of processes running on one or more host computers.

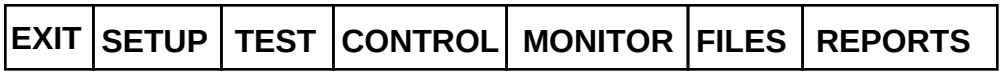
w The second possible reuse tool is the Distributed Application Monitor Tool (DAMT). This tool may help to analyze the performance of the system.

Another possible source of reuse is the Centralized Information System (CIS) of the Spacelab Data Processing Facility (SLDPF).

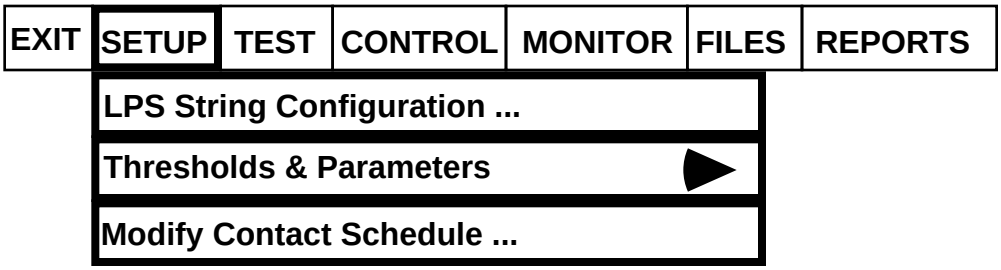
6.3.3 Oracle Screens

Oracle SQL-MENUS will be used to create a simple menu based user interface. This section presents a possible menu configuration. The purpose of these examples is to demonstrate the types of operations that can be performed via the user interface. These examples do not necessarily represent the actual look and feel of the LPS user interface. User interface screens and menus will be prototyped and reviewed with EDC operations personnel during the preliminary design phase.

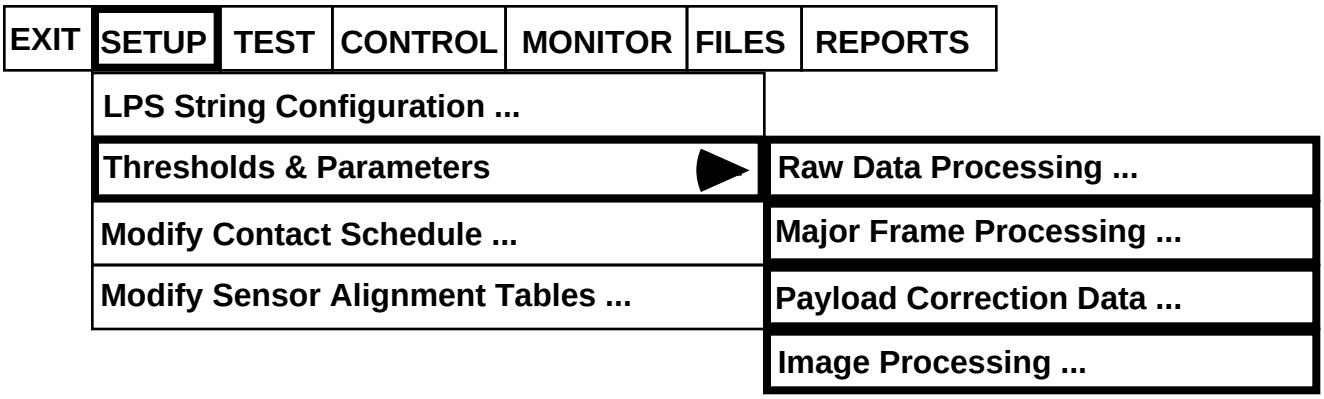
6.3.3.1 Main Menu



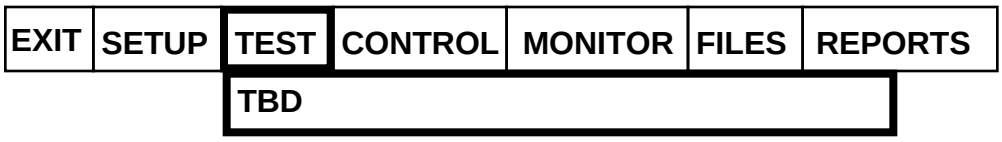
6.3.3.2 Setup Menu



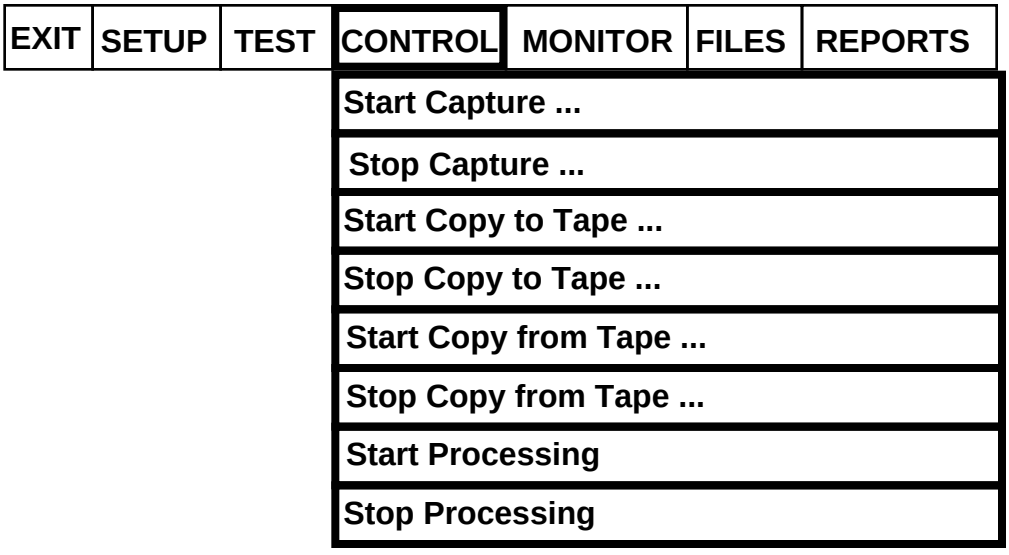
6.3.3.3 Thresholds and Parameters Menu



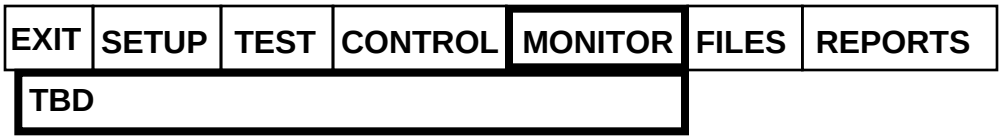
6.3.3.4 Test Menu



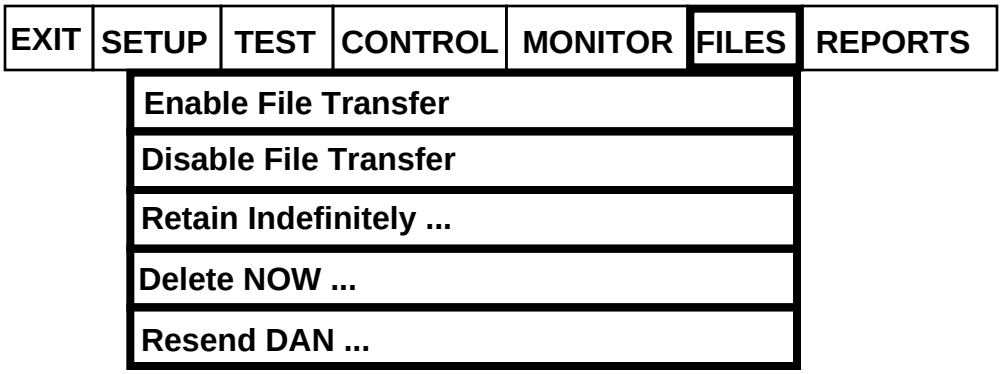
6.3.3.5 Control Menu



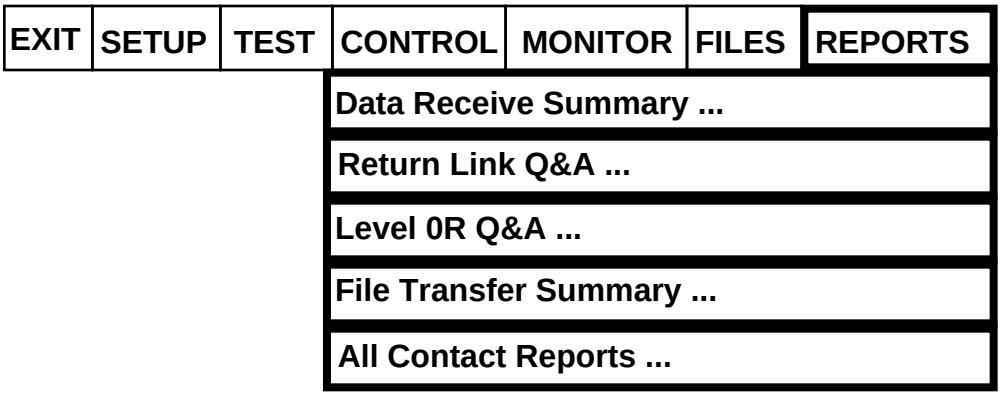
6.3.3.6 Monitor Menu



6.3.3.7 Files Menu



6.3.3.8 Reports Menu



7.0 LPS Operational Scenarios

The LPS operational scenarios represent sequences of activities performed by operations personnel as they relate to the LPS software. The scenarios may be divided into the categories of normal operations, those performed routinely to accomplish Landsat 7 data processing within the LPS, and contingency operations, those performed in response to abnormal conditions. Within each of these categories, the operational scenarios described in this section are as follows.

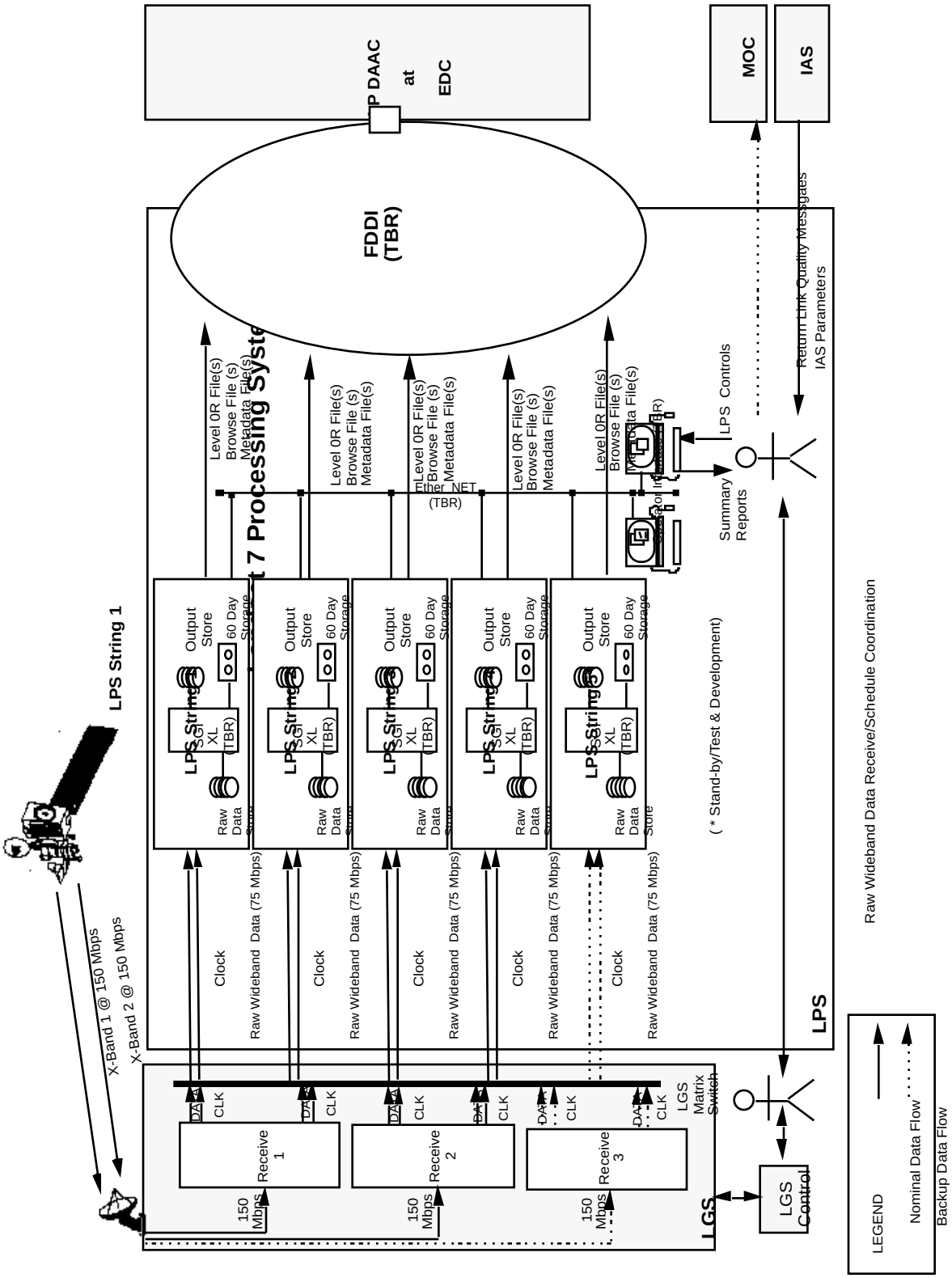
- Normal Operations
  - Receive Contact Schedule from the LGS.
  - Receive Parameters from the IAS.
  - Set Up LPS Strings for Data Capture.
  - Receive Data from the LGS.
  - Process Data to Level 0R.
  - Transfer Data to the LP DAAC.
  - Reprocess LPS Data.
  - Support Operational Training and Test.
- Contingency Operations
  - Adjust LPS Level 0R Parameters.
  - Adjust LPS Level 0R Thresholds.
  - Respond to Failure in LGS-to-LPS or LPS-to-LP DAAC Connection.
  - Respond to Exhaustion of LPS Output Storage Capacity.
  - Respond to LPS String Failure

These categories represent the majority of the activities performed by LPS operations personnel.

The operational scenarios are strongly affected by the LPS architecture presented in figure 7-1. The LPS architecture includes 5 logically independent processing strings. LPS operations use 4 processing strings at all times to support normal operations. The fifth string is available for LPS test and maintenance support, as required, as a back-up string for the 4 operational strings, and as a site for operations training and testing. The architecture also includes two workstations. One serves as the console for interface to the 4 operational strings. The second is available as an interface to

the back-up string, for LPS maintenance support, as required, and as a back-up to the operational interface workstation.

Figure 7-1
LPS Interconnect Architecture



Because of the independence between LPS strings, operations involving the 4 operational strings require that the same procedure be repeated separately on each string. However, these operations may be performed from the single interface workstation via remote login windows to each operational string.

7.1 Normal Operations

Normal LPS operations scenarios describe the sequences of operator activities performed routinely to accomplish Landsat 7 data processing within the LPS.

7.1.1 Receive Contact Schedule from the LGS

The LPS operator receives a hard-copy contact schedule from the LGS operator. The LPS operator inputs the schedule to the LPS software on each LPS string. The steps performed by the operator when a contact schedule is received are as follows.

1. Insert the new contact schedule into each LPS string database by selecting the LPS software menu option to modify contact schedules.

7.1.2 Receive Parameters from the IAS

The LPS operator receives a hard-copy list of IAS parameters from the IAS. The LPS operator inputs the parameters to the LPS software on each LPS string. The steps performed by the operator when a list of IAS parameters is received are as follows.

1. Insert the new IAS parameters into the LPS back-up string database by selecting the LPS software menu option to modify IAS parameters.
2. Execute LPS functions test for LPS functions and product verification on the back-up string.
3. Verify that outputs are correct.
4. Insert the new IAS parameters into each active string.

7.1.3 Set Up LPS Strings for Data Capture

The LPS operator verifies that the LPS configuration is correct and functional prior to each contact period. The steps in the scenario are carried out for each of the four active LPS strings but are controlled from a single workstation acting as a terminal for each string. The steps performed by the operator for LPS set up are as follows.

1. Coordinate the LGS output channel to LPS string configuration for the contact period. The LPS operator informs the LGS operator of LPS string failures so that a functioning configuration can be defined. The LGS operator implements the necessary switching to establish the agreed upon configuration.
2. Configure LPS strings to LP DAAC communication interfaces.
3. If required, start LPS user interface for each string by bringing up a window for each string and issuing a command in each to start up the LPS user interface.
4. Update the LGS output channel and LP DAAC connection information on each LPS string as required to reflect the new configuration by selecting the LPS software menu option to update LPS string configuration.
6. If required, enter/update LPS parameters for each string as described in section 7.2.1.
7. If required set LPS thresholds for each string as described in section 7.2.2.
8. Set up system monitoring on each string by selecting the LPS software menu option to set up LPS monitoring.
9. Check available disk space for minor data stores (trouble files and the LPS journal) using IRIX system commands; delete, truncate, or roll files off to tape as appropriate until sufficient disk space is available.
10. Check available disk space for raw wideband data store on each string using IRIX system commands. It is an operational decision whether existing files will be deleted or data will not be capture when insufficient disk space is available.
11. Execute LPS functions test for LPS functions and product verification on each LPS string.

7.1.4 Receive Data from the LGS

The LPS operator initiates the capture of raw wideband data by an LPS string configured to each LGS output channel which will be active during the contact period. This scenario presupposes the successful execution of the LPS string setup scenario described in section 7.1.3. The steps performed by the operator to receive data from the LGS are as follows.

1. Review contact schedule for data capture times.
2. Start data capture to disk approximately 15 seconds before scheduled acquisition of signal by selecting the LPS software menu option to start data capture for each string.
3. Verify acquisition of signal with LGS operator.
4. Monitor data receipt processes through LPS software status displays.
5. Verify loss of signal with LGS operator.
6. Stop data capture by selecting the LPS software menu option to stop data capture on each string.
7. Print and review a data receive summary report (figure 7-2) by selecting the LPS software menu option to generate the report for the contact period just captured.
8. Provide MOC with data receive summary via voice link or FAX.
9. Mount tape on drive.
10. Start copy to tape by selecting the LPS software menu option to copy to tape.
11. Label and move tape to 60-day storage.

CONTACT PERIOD		DATA VOLUME RECEIVED	
DATE REC'D	START	STOP	GBYTES APPROX SCENES
1998 123	09:15:00	09:29:03	7.8833

Figure 7-2
Draft Data Receive Summary Report Format

7.1.5 Process Data to Level 0R

The LPS operator initiates the processing to level 0R of raw wideband data on each LPS string that stores data for the contact period. The steps performed by the operator to process data to level 0R are as follows.

1. Verify that sufficient space is available in the data transfer store for output products using IRIX system commands. If retained files have been transferred to the LP-DAAC, it is an operational decision whether those files will be deleted or data processing will be delayed.
2. Start processing for selected contact by selecting the LPS software menu option to start level 0R processing.
3. Monitor the data processing function using LPS software status displays.
4. Print and review the return link quality and accounting report (figure 7-3) by selecting the LPS software option to generate this report.
5. Print and review level 0R quality and accounting report (figure 7-4) by selecting the LPS software option to generate this report.
6. Verify data are stored for LP DAAC retrieval using IRIX system commands.

LPS RETURN LINK QUALITY AND ACCOUNTING REPORT

CONTACT PERIOD
Date 1998 123 LPS String ID LPS 1
Start 09:15:00 Data Channel 4
Stop 09:29:03

Data Received in MBytes 788.01
Major Frames Received 11870
Approx #WRS Scenes Received 33
Approx Bit Error Rate 0.000001

CADU Sync Information
Search: 1
Lock: 2
Flywheel: 0
CADU Sync Marker Check: 0
CADU Sync Lock Error : 0

#CADUs received: 7572112 #CADUs w/sync errors: 25
#CADUs missing 10 #CADUs w/CRC errors: 3
#Flywheel CADUs 5 BCH on mission data zone

#CADUs corrected: 10
#CADUs uncorrected: 0
VCDU Headers: BCH on data pointer zone
#R-S correctable: 2 #CADUs corrected: 1
#R-S uncorrectable: 1 #CADUs uncorrected 0

Figure 7-3
Draft Return Link Quality and Accounting Report Format

LPS LEVEL 0R QUALITY AND ACCOUNTING REPORT

SUBINTERVAL
Date 1998 123 LPS String ID LPS 1
Start 09:15:00 Data Channel 4
Stop 09:29:03

#Major Frames in SubInterval 11870
#Major Frame ENTIRELY Filled 24
#Major Frames PARTIALLY Filled 123
Approx Bit Error Rate 0.000001

CADU Sync Information
Search: 1
Lock: 2
Flywheel: 0
CADU Sync Marker Check: 0
CADU Sync Lock Error : 0

#CADUs received: 7572112 #CADUs w/sync errors: 25
#CADUs missing 10 #CADUs w/CRC errors: 3
#Flywheel CADUs 5 BCH on mission data zone

#CADUs corrected: 10
#CADUs uncorrected: 0
VCDU Headers: BCH on data pointer zone
#R-S correctable: 2 #CADUs corrected 1
#R-S uncorrectable: 1 #CADUs uncorrected 0

Figure 7-4
Draft Level 0R Link Quality and Accounting Report Format

7.1.6 Transfer Files to the LP DAAC

Transfer of output files to the LP DAAC occurs automatically unless the override option has been set. This section describes the automatic transfer scenario. The LPS operator monitors the LPS/LP DAAC interface. The steps performed by the operator to monitor the interface are as follows.

1. Monitor interface with LP DAAC via IRIX system utilities.
2. Verify DAN sent to LP DAAC via LPS software status and journal messages.
3. Verify successful transfer of files via LPS software status and journal messages and/or transfer status of files in LPS database.
4. Print and review data transfer summary report (figure 7-5).
5. Monitor deletion of successfully transferred files via LPS software status messages and IRIX system utilities.

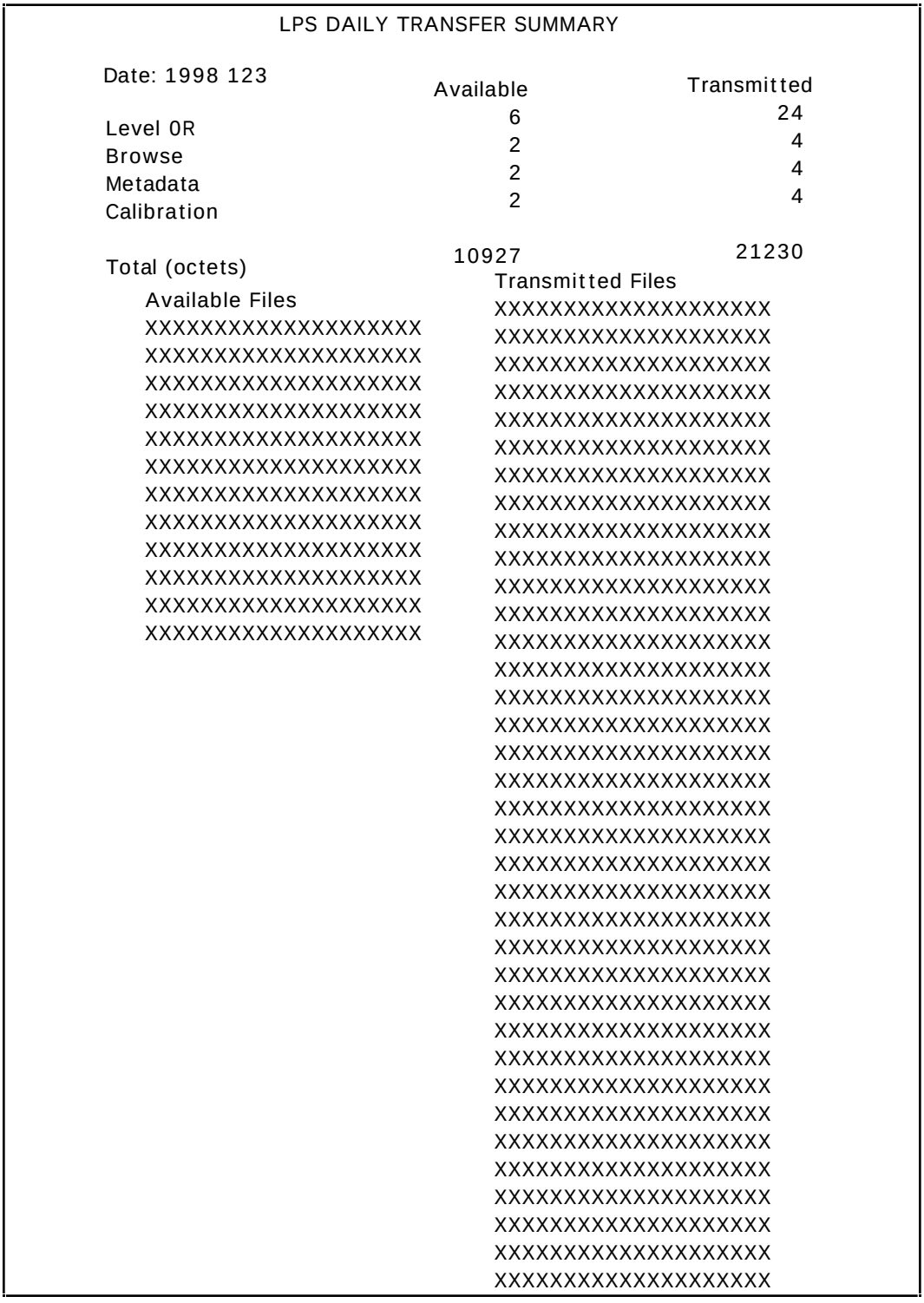


Figure 7-5
Draft Data Transfer Summary Report Format

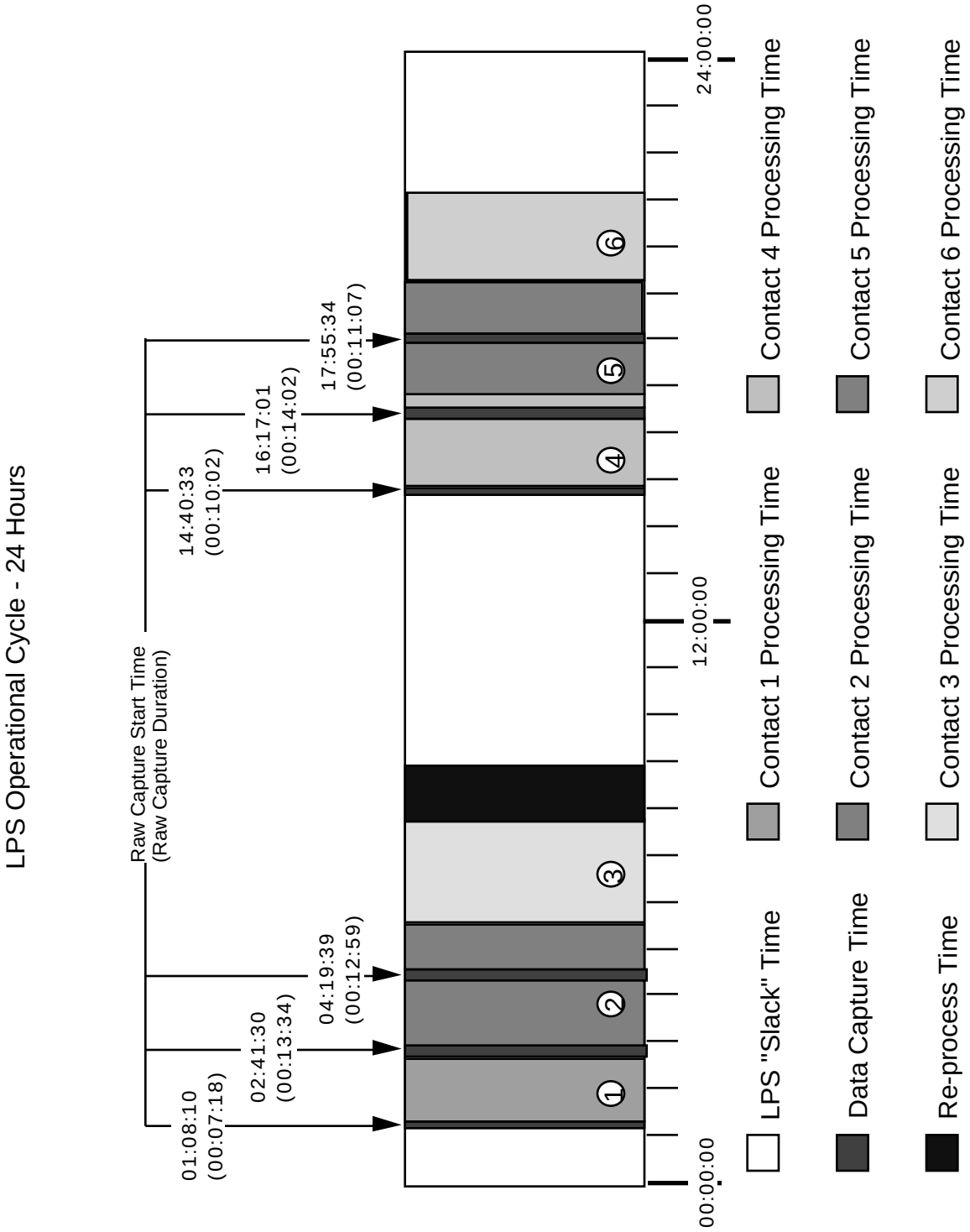
7.1.7 Reprocess LPS Data

The LPS operator initiates reprocessing for each contact period for which reprocessing has been requested by the IAS and which is still available in 60-day storage. The steps performed by the operator to reprocess data are as follows.

1. Receive reprocessing request from the IAS.
2. Verify requested data are available in 60-day storage.
3. Schedule data reprocessing time.
4. Locate all tapes for contact period in physical storage.
5. Mount each tape on a drive on a separate string.
6. Restage data to be processed (per string) by selecting LPS software menu option to restage data.
7. Return tapes to physical storage.
8. Initiate level 0R processing as described in section 7.1.5.

The LPS operational timeline, presented in figure 7-6, includes sufficient available time for reprocessing the required 10% of the LPS daily data volume.

Figure 7-6
LPS Operational Timeline



7.1.8 Support Operational Training and Test

Operational training and test support is provided by the back-up LPS string and test console with DAN transmission capabilities disabled. Test data output by the LGS may be used to support training in data capture and level 0R processing procedures. Back-up copies of LPS database contents from active strings may be used to populate a training/test database.

7.2 Contingency Operations

Contingency operations scenarios describe the sequences of operator activities performed to handle abnormal conditions during Landsat 7 data processing within the LPS. Abnormal conditions include hardware failures and storage capacity shortfalls.

7.2.1 Adjust LPS Level 0R Parameters

The LPS operator adjusts LPS Level 0R parameters whenever it is determined that Level 0R output can be increased by adjusting the processing parameters listed in section 6.1.5. The steps performed by the operator to adjust Level 0R parameters are as follows.

1. Determine desired values for parameters.
2. Adjust level 0R parameters on the LPS back-up string by selecting the LPS software menu options that adjust level 0R processing parameters.
3. Execute LPS functions test for LPS functions and product verification on the back-up string.
4. Verify that outputs are correct.
5. Adjust level 0R parameters on each LPS operational string.

7.2.2 Adjust LPS Level 0R Thresholds

The LPS operator can adjust LPS Level 0R thresholds described in section 6.1.5 whenever excessive noise in raw wideband data causes a proliferation of alarms and alerts. The steps performed by the operator to adjust level 0R thresholds are as follows.

1. Determine desired values for thresholds for each error type.
2. Adjust level 0R thresholds on each LPS string by selecting the LPS software menu options that adjust level 0R thresholds.

7.2.3 Respond to Failure in LGS/LPS Interface

Either the LPS or LGS operator may detect a LGS/LPS interface failure. The steps performed by the LPS operator to respond to a failure in the LGS/LPS interface are as follows.

1. If the LPS operator has detected the failure, then notify the LGS operator.
2. If the failure occurs while data is being captured, notify the MOC and the LP-DAAC of the data loss.
3. Continue level 0R processing on any data captured prior to the failure, including a partial contact period.
4. When the LGS/LPS interface has been restored, coordinate with the LGS operator to test the interface by capturing a test data set sent from the LGS.
5. Verify the successful transmission of the test data set.

7.2.4 Respond to Failure in LPS/LP-DAAC Interface

The LPS operator may detect an LPS/LP-DAAC interface failure while monitoring LPS disk space available for transfer storage. The LPS operator may also be notified of the failure by the LP-DAAC operator. The steps performed by the operator to handle this condition are as follows.

1. If the LPS operator has detected the failure, then notify the LP-DAAC operator.
2. Do not perform additional level 0R processing.
3. Continue to capture data and to copy the captured data to the 60-day store; ensure sufficient capture capacity by deleting on-line captured data sets copied to the 60-day store.
4. Resume level 0R processing when the interface is restored.

5. Handle processing for the back-log of data in the 60-day store in the same way as reprocessing as described in section 7.1.7.

The LPS operational timeline, presented in figure 7-6, includes sufficient excess time to support the additional processing load imposed by the suspension of level 0R processing during the interface failure.

7.2.5 Respond to Exhaustion of LPS Output Storage Capacity

The LPS operator detects that LPS output storage capacity has been exceeded while monitoring the output storage available on LPS strings. The steps performed by the operator to handle this condition are as follows.

1. Continue receiving data from the LGS as scheduled.
2. Do not initiate level 0R processing on any received data until output storage is available. Coordinate with LP DAAC operator in regard to the LP DAAC schedule for transferring retained files.

7.2.6 Respond to LPS String Failure

LPS string failures occur whenever the LPS string Data Process HWCI or any of its peripheral HWCIs fails. The failure of any part of the string is treated identically to a failure of the whole. The steps performed by the operator to handle LPS string failures are as follows.

1. Coordinate with the LGS operator to switch the LGS output channel received by the failed string to the back-up string.
2. Coordinate with LP DAAC operator to determine new LP DAAC connection.
3. Set up the back-up LPS string to replace the failed operational string, following the scenario described in section 7.1.3.
4. Initiate data capture and level 0R processing of captured data on the back-up string.
4. Notify LPS maintenance personnel of the failure.

4. Notify the LP DAAC operator of the failure and cancel any DANs outstanding from the failed string.

5. Reprocess data that was processed by the failed string but not yet transferred to the LP DAAC from the 60-day store to any string with excess processing capacity following the scenario described in section 7.1.7.

6. Stage data captured by the failed string and copied to the 60 day store but not yet processed, then stage the data to any string with excess processing capacity following the scenario for reprocessing described in section 7.1.7.

7. When the failed LPS string has been restored, notify the LP DAAC operator that the restored string will be made operational.

8. Verify that all data available has been transferred to the LP DAAC from the back-up string.

9. Coordinate with the LGS operator to switch the LGS output channel routed to the back-up string to the restored operational string.

10. Set up the restored string to replace the back-up string, following the scenario described in section 7.1.3.

11. Initiate data capture and level 0R processing of captured data on the restored string.

12. Initiate level 0R processing for any captured data on the restored string's disks that have not yet been processed by another string.

The LPS operational timeline, presented in figure 7-6, includes sufficient excess time to support the additional processing load imposed on the remaining operational strings by the need to process data from the failed string.

Appendix A - Requirements Traceability

This appendix presents the LPS requirements traceability. The first table shows the mapping between F&PS requirements and the lowest level processes in the data flow diagrams. The second table shows the mapping between the lowest level processes in the data flow diagrams and the F&PS requirements. Note that some system and performance requirements which apply to every processes are not shown in the tables.

A.1 System to Software Requirements Traceability

Requirement Number	Process Number	Process Name
3.1. 4	2	Synchronize CCSDS Frame
	2	Process CCSDS Grade 3
	2	Decode BCH
	2	Generate Return Link QA Report
	2	Validate RDP Parameters
	2	Compute BER
	2.2	Perform SCLF Sync
	2.2	Align Bytes
	2.2	Deinvert Data
	2.2	Perform PN Decode
	2.3	Perform CRC Check
	2.3	Perform RS_EDAC Check
	2.3	Discard Fill CADUs
	2	Annotate VCID Change
3.1. 5	3.5	Align Bands
	5	Generate Band File
	5.2	Reduce Image by Wavelets
	5.2	Reduce Image by Subsamples
	6	Generate Metadata
3.1. 6	2	Decode BCH
	2	Generate Return Link QA Report
	2	Validate RDP Parameters
	2	Compute BER
	2.3	Perform CRC Check
	2.3	Perform RS_EDAC Check
3.1. 7	3	Generate Level 0R QA Report
3.1. 8	6	Process LPS Directive
3.1.10	6.1	LPS System Control
3.1.10.1	6.1	LPS System Control
3.1.10.3	2	Decode BCH
	2	Validate RDP Parameters
	2	Compute BER
	2.2	Perform SCLF Sync
	2.3	Perform CRC Check
	2.3	Perform RS_EDAC Check
	6	Report LPS Status
3.1.10.4	6	Monitor System Faults

3.1.10.5	6	Monitor System Faults
3.1.11	2	Synchronize CCSDS Frame
	2	Validate RDP Parameters
	6	Report LPS Status
	6	Process LPS Directive
3.1.12	2	Decode BCH
	2	Validate RDP Parameters
	2	Compute BER
	2.2	Perform SCLF Sync
	2.3	Perform CRC Check
	2.3	Perform RS_EDAC Check
	6	Report LPS Status
3.1.14	6.1	Modify LPS Configuration
3.1.19	6	Report LPS Status
3.2.2	7	Send DAN
	7	Transfer Files
3.2.4	6	Process LPS Directive
3.3.1. 1	1	Receive Raw Wideband Data
	1	Delete Raw Wideband Data
3.3.1. 2	1	Receive Raw Wideband Data
	1	Delete Raw Wideband Data
3.3.1. 3	1	Receive Raw Wideband Data
	1	Delete Raw Wideband Data
3.3.1. 4	1	Receive Raw Wideband Data
	1	Delete Raw Wideband Data
3.3.1. 7	1	Save Raw Wideband Data
3.3.1. 8	1	Save Raw Wideband Data
3.3.1. 9	1	Restage Raw Wideband Data
3.3.1.12	1	Receive Raw Wideband Data
3.3.2. 1	2.3	Perform CRC Check
	2.3	Perform RS_EDAC Check
	2.3	Discard Fill CADUs
3.3.2. 3	2.2	Perform SCLF Sync
	2.2	Align Bytes
	2.2	Deinvert Data
3.3.2. 4	2.2	Perform SCLF Sync
3.3.2. 5	2.2	Deinvert Data
3.3.2. 6	2.2	Perform SCLF Sync
3.3.2. 7	2.2	Perform PN Decode
3.3.2. 8	2.3	Perform CRC Check
	2.3	Perform RS_EDAC Check
3.3.2. 9	2	Decode BCH
3.3.2. 9.1	2	Decode BCH
3.3.2.10	2	Decode BCH
3.3.2.11	3.4	Determine Subintervals
	2	Annotate VCID Change
3.3.2.12	2.3	Discard Fill CADUs
3.3.2.13	2	Decode BCH
	2	Compute BER
	2.2	Perform SCLF Sync
	2.3	Perform CRC Check
	2.3	Perform RS_EDAC Check
	3.4	Extract and Collect VCDU Quality and Accounting
	3.4	Collect VCDU Quality and Accounting
3.3.2.14	3.4	Identify Major Frames
3.3.2.15	3.4	Identify Major Frames
3.3.2.16	3.5	Deinterleave and Reverse Bands

3.3.2.17	3.5	Deinterleave and Reverse Bands
3.3.2.18	3.5	Deinterleave and Reverse Bands
3.3.2.19	3.5	Deinterleave and Reverse Bands
	3.5	Align Bands
3.3.2.20	3.6	Extract MSCD Data
3.3.2.21	3.6	Extract Calibration Data
3.3.2.22	3.5	Align Bands
3.3.2.23	3.4	Determine Subintervals
3.3.2.24	3.5	Deinterleave and Reverse Bands
	3.5	Align Bands
	5	Generate Band File
3.3.2.25	3.6	Create MSCD File
	3.6	Extract MSCD Data
	4	Create PCD File
	5	Generate Band File
	5.2	Reduce Image by Wavelets
	5.2	Reduce Image by Subsamples
3.3.2.26	3	Identify VCDUs
	3.4	Identify Major Frames
	3.4	Determine Subintervals
	3.4	Extract and Collect VCDU Quality and Accounting
	3.4	Collect VCDU Quality and Accounting
3.3.2.27	4	Extract Major Frame Info
	5	Generate Band File
3.3.2.28	3.5	Align Bands
	5	Generate Band File
3.3.2.29	4.4	Compute Position MJF Time
3.3.3.1	5.2	Reduce Image by Wavelets
	5.2	Reduce Image by Subsamples
3.3.3.2	5.2	Reduce Image by Wavelets
	5.2	Reduce Image by Subsamples
3.3.3.3	5.2	Reduce Image by Wavelets
	5.2	Reduce Image by Subsamples
3.3.3.4	5.2	Reduce Image by Subsamples
3.3.3.5	5.2	Reduce Image by Wavelets
	5.2	Reduce Image by Subsamples
3.3.4. 1	4.2	Extract Info Word
3.3.4. 2	3	Extract PCD
	4.3	Assemble Minor Frames
	4.3	Assemble Major Frames
3.3.4. 3	4.3	Assemble Major Frames
3.3.4. 4	4	Create PCD File
3.3.4. 5	4	Create PCD File
	4.3	Build PCD Cycles
3.3.4. 6	4.3	Build PCD Cycles
3.3.4. 7	4.4	Determine WRS Scene Coordinates
3.3.4. 8	5.4	Generate Cloud Cover Assessment
	5.4	Collect Scene Data
3.3.4. 9	5.4	Generate Cloud Cover Assessment
3.3.4.10	5.4	Generate Cloud Cover Assessment
3.3.4.11	6	Generate Metadata
3.3.4.12	6	Generate Metadata
3.3.5.1	7	Send DAN
3.3.5.2	7	Transfer Files
3.3.5.3	7	Receive DTA
3.3.5.4	7	Delete LPS Files
	7	Retain LPS Files

3.3.5.5	7	Delete LPS Files
3.3.5.6	7	Retain LPS Files
3.3.5.7	7	Generate Transfer Summary Report
3.3.6.1	2	Validate RDP Parameters
	6	Process LPS Directive
	6.1	Modify Contact Schedule
	3	Validate MFP Parameters
	5	Validate IDP Parameters
3.3.6.2	2	Decode BCH
	2	Generate Return Link QA Report
	2	Compute BER
	2.2	Perform SCLF Sync
	2.3	Perform CRC Check
	2.3	Perform RS_EDAC Check
3.3.6.3	6	Process LPS Directive
	3	Generate Level 0R QA Report
	6	Display or Print LPS Report
3.3.6.4	6	Process LPS Directive
	6	Display or Print LPS Report
3.3.6.5	6	Process LPS Directive
	6	Display or Print LPS Report
3.3.6.6	4.3	Assemble Minor Frames
	5.4	Generate Cloud Cover Assessment
	5.4	Collect Scene Data
	6	Report LPS Status
	6	Process LPS Directive
	5	Validate IDP Parameters
3.3.6.7	4.3	Assemble Minor Frames
	5.4	Generate Cloud Cover Assessment
	5.4	Collect Scene Data
	6	Report LPS Status
	5	Validate IDP Parameters
3.3.6.8	1	Receive Raw Wideband Data
	1	Restage Raw Wideband Data
	1	Save Raw Wideband Data
	2	Synchronize CCSDS Frame
	2	Validate RDP Parameters
	6	Process LPS Directive
	7	Delete LPS Files
	7	Retain LPS Files
3.3.6.9	1	Receive Raw Wideband Data
	2	Synchronize CCSDS Frame
	6	Process LPS Directive
	6.1	LPS System Control
4.1.6	2	Synchronize CCSDS Frame
	2	Process CCSDS Grade 3
	2	Decode BCH
	2	Generate Return Link QA Report
	2	Validate RDP Parameters
	2	Compute BER
	2.2	Perform SCLF Sync
	2.2	Align Bytes
	2.2	Deinvert Data
	2.2	Perform PN Decode
	2.3	Perform CRC Check
	2.3	Perform RS_EDAC Check
	2.3	Discard Fill CADUs
	2	Annotate VCID Change

4.3.3	**2**	**Synchronize CCSDS Frame**
4.3.4	**5.2**	**Reduce Image by Wavelets**
	5.2	**Reduce Image by Subsamples**
4.3.5	**4.4**	**Determine WRS Scene Coordinates**

A.2 Software to System Requirements Traceability

Process Number	Process Name	Requirement Number
1	Receive Raw Wideband Data	3.3.1. 1 3.3.1. 2 3.3.1. 4 3.3.6.8 3.3.6.9 3.3.1.12 3.3.1. 3
1	Restage Raw Wideband Data	3.3.1. 9 3.3.6.8
1	Save Raw Wideband Data	3.3.1. 7 3.3.1. 8 3.3.6.8
1	Delete Raw Wideband Data	3.3.1. 1 3.3.1. 4 3.3.1. 2 3.3.1. 3
1	Generate Data Receive Summary Report	
2	Generate Return Link QA Report	3.1. 4 4.1.6 3.3.6.2 3.1. 6
2	Decode BCH	3.1. 4 3.1. 6 3.1.12 3.1.10.3 3.3.2. 9.1 3.3.2.13 4.1.6 3.3.6.2 3.3.2.10 3.3.2. 9
2	Annotate VCID Change	3.1. 4 4.1.6 3.3.2.11
2	Process CCSDS Grade 3	3.1. 4 4.1.6
2	Validate RDP Parameters	3.1. 4 3.1. 6 4.1.6 3.3.6.8 3.3.6.1 3.1.12 3.1.11 3.1.10.3
2	Compute BER	3.1. 4 3.3.6.2 3.3.2.13 3.1. 6 4.1.6

		3.1.12
		3.1.10.3
2	Synchronize CCSDS Frame	3.1. 4
		4.3.3
		4.1.6
		3.1.11
		3.3.6.9
		3.3.6.8
2.2	Perform PN Decode	3.1. 4
		3.3.2. 7
		4.1.6
2.2	Deinvert Data	3.1. 4
		4.1.6
		3.3.2. 5
		3.3.2. 3
2.2	Align Bytes	3.1. 4
		3.3.2. 3
		4.1.6
2.2	Perform SCLF Sync	3.1. 4
		4.1.6
		3.3.6.2
		3.3.2.13
		3.1.12
		3.3.2. 3
		3.1.10.3
		3.3.2. 6
		3.3.2. 4
2.3	Perform CRC Check	3.1. 4
		4.1.6
		3.3.2.13
		3.3.6.2
		3.3.2. 8
		3.1.12
		3.3.2. 1
		3.1.10.3
		3.1. 6
2.3	Discard Fill CADUs	3.1. 4
		3.3.2. 1
		4.1.6
		3.3.2.12
2.3	Perform RS_EDAC Check	3.1. 4
		4.1.6
		3.3.6.2
		3.3.2.13
		3.3.2. 8
		3.1. 6
		3.1.10.3
		3.1.12
		3.3.2. 1
3	Validate MFP Parameters	3.3.6.1
3	Generate Level 0R QA Report	3.1. 7
		3.3.6.3
3	Extract PCD	3.3.4. 2
3	Identify VCDUs	3.3.2.26
3	Parse Major Frame	
3	Extract Calibration and MSCD	
3	Collect Quality and Accounting	
3	Generate Band Data	

3.4	Determine Subintervals	3.3.2.11
		3.3.2.23
		3.3.2.26
3.4	Extract Major Frame Time	
3.4	Collect VCDU Quality and Accounting	3.3.2.13
		3.3.2.26
3.4	Identify Major Frames	3.3.2.14
		3.3.2.15
		3.3.2.26
3.5	Deinterleave and Reverse Bands	3.3.2.16
		3.3.2.17
		3.3.2.24
		3.3.2.18
		3.3.2.19
3.5	Align Bands	3.1. 5
		3.3.2.24
		3.3.2.19
		3.3.2.22
		3.3.2.28
3.6	Create MSCD File	3.3.2.25
3.6	Extract Calibration Data	3.3.2.21
3.6	Extract MSCD Data	3.3.2.20
		3.3.2.25
3.6	Create Calibration File	
4	Calculate Scene Info	
4	Create PCD File	3.3.2.25
		3.3.4. 4
		3.3.4. 5
4	Extract Major Frame Info	3.3.2.27
4	Assemble PCD Cycles	
4	Determine PCD Info Word	
4	Validate PCD Parameters	
4.2	Determine Majority Info Word	
4.2	Extract Info Word	3.3.4. 1
4.3	Assemble Major Frames	3.3.4. 2
		3.3.4. 3
4.3	Build PCD Cycles	3.3.4. 5
		3.3.4. 6
4.3	Assemble Minor Frames	3.3.4. 2
		3.3.6.7
		3.3.6.6
4.4	Calculate Sun Position	
4.4	Determine WRS Scene Coordinates	3.3.4. 7
		4.3.5
4.4	Report Scene Info	
4.4	Compute Horizontal Display Shift	
4.4	Compute Position MJF Time	3.3.2.29
4.4	Compute Latitude And Longitude	
5	Perform ACCA	
5	Validate IDP Parameters	3.3.6.1
		3.3.6.6
		3.3.6.7
5	Generate Band File	3.1. 5
		3.3.2.28
		3.3.2.27
		3.3.2.25
		3.3.2.24
5	Generate Browse File	

5.2	**Reduce Image by Wavelets**	**3.1. 5**
		4.3.4
		3.3.3.5
		3.3.3.1
		3.3.3.2
		3.3.3.3
		3.3.2.25
5.2	**Reduce Image by Subsamples**	**3.1. 5**
		4.3.4
		3.3.3.5
		3.3.3.4
		3.3.3.3
		3.3.3.2
		3.3.3.1
		3.3.2.25
5.4	**Collect Scene Data**	**3.3.4. 8**
		3.3.6.7
		3.3.6.6
5.4	**Generate Cloud Cover Assessment**	**3.3.4. 8**
		3.3.4.10
		3.3.6.7
		3.3.6.6
		3.3.4. 9
6	**Generate Metadata**	**3.1. 5**
		3.3.4.11
		3.3.4.12
6	**Report LPS Status**	**3.1.10.3**
		3.1.12
		3.1.19
		3.1.11
		3.3.6.7
		3.3.6.6
6	**Process LPS Directive**	**3.1. 8**
		3.1.11
		3.3.6.9
		3.3.6.8
		3.3.6.6
		3.3.6.5
		3.3.6.4
		3.3.6.3
		3.3.6.1
		3.2.4
6	**Display or Print LPS Report**	**3.3.6.3**
		3.3.6.4
		3.3.6.5
6	**Monitor System Faults**	**3.1.10.4**
		3.1.10.5
6.1	**LPS System Control**	**3.1.10.1**
		3.1.10
		3.3.6.9
6.1	**Modify LPS Configuration**	**3.1.14**
6.1	**Modify Contact Schedule**	**3.3.6.1**
7	**Control Send DAN**	
7	**Generate Transfer Summary Report**	**3.3.5.7**
7	**Retain LPS Files**	**3.3.5.4**
		3.3.5.6
		3.3.6.8
7	**Transfer Files**	**3.2.2**

		3.3.5.2
7	**Delete LPS Files**	**3.3.5.4**
		3.3.5.5
		3.3.6.8
7	**Receive DTA**	**3.3.5.3**
7	**Generate DAN**	
7	**Send DAN**	**3.2.2**
		3.3.5.1

Appendix B - Data Dictionary

ACCA (data ,)

= Scene_Id +
CCA_Quadrant1_Score +
CCA_Quadrant2_Score +
CCA_Quadrant3_Score +
CCA_Quadrant4_Score +
CCA_Aggregate_Score.
*
Five percentage scores indicating the amount of cloud coverage including one score for each quadrant and one aggregate score for an entire WRS scene. The metadata contains the scene id and sub interval id to uniquely identify the scores.
*
---------- end definition ----------

ACCA_Acct (data ,)

= Sub_Intv_Id +
CCA_Method +
ACCA.
*
File containing the CCA method and the ACCA scores. This information is stored in the datastore IDP_Acct.
*
---------- end definition ----------

Actual_Center_Coords (data ,)

= Sub_Intv_Id +
Longitude +
Latitude +
Scene_Center_Time.
*
The computed scene center position and corresponding time. There is one set of information for each identified scene.
*
---------- end definition ----------

Actual_Time (data ,)

= Real.
*
The actual major frame time to 1/16th of a millisecond.
*
---------- end definition ----------

```
Address                    ( data      , primitive   )

  = *
    Internal location of data.
    *.
     ---------- end definition ----------


ADS                        ( data      ,          )

  = Byte.
    *
    The result of the sample of up to four Attitude Displacement
    Sensor related temperatures.  Each sample is inserted into two
    consecutive bytes of the PCD minor frame.  There will be eight

    ADS-related bytes in total.
    *
     ---------- end definition ----------


Aligned_Bands              ( data      ,          )

  = Sub_Intv_Id +
    [
     Fmt1_Align_Data |
     Fmt2_Align_Data
    ].
    *
    Band data after all alignments have been performed. This is
    done according to the Sensor_Alignment_Info.
    *
     ---------- end definition ----------


Aln_CADU                   ( data      ,          )

  = Contact_Id +
    Sync +
    CADU_Bytes +
    Sync_Annotation
    *
    A CADU which has been aligned on a byte boundary with the
    sync quality indicators

    NOTE:
    The contact ID is not part of the annotation, but simply
    shown as information necessary to associate this CADU with
    a Contact_Id.
    *
     ---------- end definition ----------
```

```
Aln_Inver_CADU                ( data      ,         )

  = Contact_Id +
    Inverted_Sync +
    Inverted_CADU_Bytes +
    Sync_Annotation
    *
    An inverted CADU which has been aligned on a byte boundary
    with the sync quality indicators

    NOTE:
    The contact ID is not part of the annotation, but simply
    shown as information necessary to associate this CADU with
    a Contact_Id.
    *
     ---------- end definition ----------


Ann_CADU                      ( data      ,         )

  = Contact_Id +
    Sync +
    PN_Decoded_CADU_Bytes +
    Sync_Annotation
    *
    A CADU that includes the sync marker, the bytes that have been PN
    decoded and the frame sync quality annotations.

    NOTE:

    The contact ID is not part of the annotation, but simply
    shown as information necessary to associate this CADU with
    a Contact_Id.
    *
     ---------- end definition ----------


Ann_VCDU                      ( data      ,         )

  = Contact_Id +
    Sync +
    VCDU_Hdr_Bytes +
    [
     VCDU_Data |
     BCH_Corrected_Data
    ] +
    VCDU_Trailer +
    Sync_Annotation +
    CRC_Annotation +
    RS_Annotation +
    BCH_Annotation +
    VCID_Change_Flag.
    *
```

```
    A VCDU that has completed all of the error detection and correction
    processes and has been checked for a change in VCID along with the
    data quality annotations and a flag to indicate a change in the VCID.

    NOTE:
    The Contact ID and End_Of_Contact_Flag are not part of the
    annotation, but simply shown as information necessary to
    associate this VCDU with a Contact_Id and the end of contact.
    *
     ---------- end definition ----------


Ann_VCDU_Collection              ( store     ,         )

  = Rel_VCDU_Cnt +
    0{
     Ann_VCDU +
     Num_Missing_VCDUs
    }Rel_VCDU_Cnt +
    Exp_VCDU_Ctr +
    Exp_Mnf_Ctr +
    Mjf_CADU_Seq_Err_Cnt +
    Mjf_CADU_Fly_Cnt +
    Mnf_Ctr_Err
    *
    The storage for annotated VCDUs during major frame creation.
    *
     ---------- end definition ----------


App_Data_Band1                   ( data      ,         )

  = {
     Byte
    } +
    Status_Info +
    Drift_Time.
    *
    Format 1 Scene Data for band 1 appended with the
    Status_Info and Drift_Time.
    *
     ---------- end definition ----------


App_Data_Band2                   ( data      ,         )


  = {
     Byte
    } +
    Status_Info +
    Drift_Time.
    *
    Format 1 Scene Data for band 2 appended with the
```

```
    Status_Info and Drift_Time.
    *
     ---------- end definition ----------


App_Data_Band3              ( data      ,        )

  = {
     Byte
    } +
    Status_Info +
    Drift_Time.
    *
    Format 1 Scene Data for band 3 appended with the
    Status_Info and Drift_Time.
    *
     ---------- end definition ----------


App_Data_Band4              ( data      ,        )

  = {
     Byte
    } +
    Status_Info +
    Drift_Time.
    *
    Format 1 Scene Data for band 4 appended with the
    Status_Info and Drift_Time.
    *
     ---------- end definition ----------


App_Data_Band5              ( data      ,        )

  = {
     Byte
    } +
    Status_Info +
    Drift_Time.
    *
    Format 1 Scene Data for band 5 appended with the
    Status_Info and Drift_Time.
    *
     ---------- end definition ----------


App_Data_Band6              ( data      ,        )

  = {
     Byte
    } +
    Status_Info +
    Drift_Time.
```

```
    *
    Format 1 or 2 Scene Data for band 6 appended with the

    Status_Info and Drift_Time.
    *
     ---------- end definition ----------


App_Data_Band7                ( data      ,         )

  = {
     Byte
    } +
    Status_Info +
    Drift_Time.
    *
    Format 2 Scene Data for band 7 appended with the
    Status_Info and Drift_Time.
    *
     ---------- end definition ----------


App_Data_Pan                  ( data      ,         )

  = {
     Byte
    } +
    Status_Info +
    Drift_Time.
    *
    Format 2 Pan Data appended with the Status_Info and
    Drift_Time.
    *
     ---------- end definition ----------


App_Format_1_Bands            ( data      ,         )

  = App_Data_Band1 +
    App_Data_Band2 +
    App_Data_Band3 +
    App_Data_Band4 +
    App_Data_Band5 +
    App_Data_Band6.
    *
    Data for bands 1 through 6 appended with the drift time.
    *
     ---------- end definition ----------


App_Format_2_Bands            ( data      ,         )

  = App_Data_Band6 +
    App_Data_Band7 +
```

App_Data_Pan.
*
Data for bands 6, 7 and Pan appended with the drift time.
*
---------- end definition ----------

Approx_Data_Received (data ,)

= Natural.
*
The approximate amount of wideband data received in megabytes.
*
---------- end definition ----------

Approx_ETM_Count (data ,)

= Natural.
*
The approximate number of ETM+ scenes.
*
---------- end definition ----------

Approx_Major_Frame_Count (data ,)

= Natural.
*
The approximate number of major frames.
*
---------- end definition ----------

Att_Lower_Bounds (data ,)

= Real.
*
The lowest valid value of any attitude component. (TBD)
*
---------- end definition ----------

Att_Upper_Bounds (data ,)

= Real.
*
The highest valid value of any attitude component. (TBD)
*
---------- end definition ----------

```
Attitude                    ( data      ,          )

  = EPA1 +
    EPA2 +
    EPA3 +
    EPA4.
    *
    The components of the quaternion which describes the position
    of the spacecraft are located in each PCD Major Frame.PCD Minor
    Frame 0-15.Word 72.
    *
     ---------- end definition ----------


Available_Browse_File_Count         ( data      ,          )

  = Natural
    *
    Count of available Browse files.
    *
     ---------- end definition ----------


Available_L0R_File_Count            ( data      ,          )

  = Natural
    *
    Count of available L0R files.

    *
     ---------- end definition ----------


Available_LPS_File_Names            ( data      ,          )

  = { Contact_File_Names }.

    *
    The names of LPS output files which are available for transfer
    to the LP DAAC but for which a DAN has not yet been sent.
    *
     ---------- end definition ----------


Available_Metadata_Count            ( data      ,          )

  = Natural
    *
    Count of available Metadata files.
    *
     ---------- end definition ----------


Available_Retention_Space           ( data      ,          )
```

```
  = Natural.
   *
   This object represents the number of units of volume of disk
   space which are currently available to be used for on-line
   retention of LPS output files.
   *
    ---------- end definition ----------


Band1_File_Type                ( data      , discrete    )

  = "Band1".
   *
   A string identifying the Band 1 file.
   *
    ---------- end definition ----------


Band2_File_Type                ( data      , discrete    )

  = "Band2".
   *
   A string identifying the Band 2 file.
   *
    ---------- end definition ----------


Band3_File_Type                ( data      , discrete    )

  = "Band3".
   *
   A string identifying the Band 3 file.
   *
    ---------- end definition ----------


Band4_File_Type                ( data      , discrete    )

  = "Band4".
   *

   A string identifying the Band 4 file.
   *
    ---------- end definition ----------


Band5_File_Type                ( data      , discrete    )

  = "Band5".
   *
   A string identifying the Band 5 file.
   *
    ---------- end definition ----------
```

```
Band6_Det_Data                ( data      ,          )

  = 6313{Byte}6313.
   *
   Aligned detector data for band 6.
   *
    ---------- end definition ----------


Band6_File_Type               ( data      , discrete    )

  = "Band6".
   *
   A string identifying the Band 6 file.
   *
    ---------- end definition ----------


Band7_File_Type               ( data      , discrete    )

  = "Band7".
   *
   A string identifying the Band 7 file.
   *
    ---------- end definition ----------


Band_Acct                     ( data      ,          )

  = Sub_Intv_Id +
   Bands_Present +
   Band_File_Names +
   Band_Gains +
   Gain_Change_Flag
   *
   A datastore containing the sub-interval identifier, the
   band numbers for which band files were created and the
   band file names.
   *
    ---------- end definition ----------


Band_Det_Data                 ( data      ,          )

  = 6313{Byte}6313.
   *
   The aligned band data array for bands 1 through 5 and 7.
   Data for each detector consists of 6313 bytes.
   *
    ---------- end definition ----------
```

```
Band_File                    ( store     ,         )


  = [
     App_Format_1_Bands |
     App_Format_2_Bands
    ].
    *
    A set of files of deinterleaved data with one file per
    band.  Each file has been appended with the Status_Info
    and Drift_Time.  Format 1 bands are 1-6; format 2 bands
    are 6, 7 and Pan.
    *
     ---------- end definition ----------


Band_File_Name               ( data      ,         )

  = File_Location +
    "L7_" +
    MF_Start_Time +
    "_" +
    Band_File_Type +
    "_" +
    File_Version_Number.
    *
    The band file name.
    *
     ---------- end definition ----------


Band_File_Names              ( data      ,         )

  = [
     Band_File_Names_Fmt1 |
     Band_File_Names_Fmt2
    ].
    *
    There are six band file names for format 1.
    There are three band file names for format 2.
    *
     ---------- end definition ----------


Band_File_Names_Fmt1         ( data      ,         )

  = 6{
     Band_File_Name
    }6.
    *
    The band file names for format 1 data
    *
     ---------- end definition ----------
```

```
Band_File_Names_Fmt2            ( data      ,       )

  = 3{
    Band_File_Name
   }3.
   *
   The band file names for format 2 data
   *
    ---------- end definition ----------


Band_File_Type                  ( data      ,       )


  = [Band1_File_Type
    |Band2_File_Type
    |Band3_File_Type
    |Band4_File_Type
    |Band5_File_Type
    |Band6_File_Type
    |Band7_File_Type
    |Band_Pan_File_Type]
   *
   The type of band file generated.
   *
    ---------- end definition ----------


Band_Gain                       ( data      ,       )

  = Bit.
   *
   1 bit of word 8 of the PCD/Status field of the VCDU.  This
   applies to bands 1 through 5 and band 7.
   Low gain = "0". High gain = "1".
   *
    ---------- end definition ----------


Band_Gain6                      ( data      ,       )

  = Bit.
   *
   Band gain 6 only.  In the PCD/Status field of the VCDU, word 8 bit 6 defines the band gain for
   Format_Id = 1 and bit 7 for Format_Id = 2.  Low gain = "0". High gain = "1".
   *
    ---------- end definition ----------


Band_Gain_Pan                   ( data      ,       )

  = Bit.
   *
```

```
    Band gain PAN bit. Word 7 bit 8 of PCD/Status Data.
    Low gain = "0". High gain = "1".
    *
     ---------- end definition ----------


Band_Gains                  ( data      ,         )

  = Band_Gain
    + Band_Gain6
    + Band_Gain_Pan
    *
    Description:
    Band gains for LPS bands (1 to 8).
    *
     ---------- end definition ----------


Band_Num                    ( data      ,         )

  = Natural
    *
    The band number of the Aligned_Bands.Band_Det_Data.
    *.
     ---------- end definition ----------


Band_Pan_File_Type              ( data      , discrete    )


  = "Band_Pan".
    *
    A string identifying the Pan Band file.
    *
     ---------- end definition ----------


Band_Store                  ( store     ,         )

  = {Aligned_Bands}.
    *
    A datastore that accumulates the Aligned_Bands data
    for Band Processing until the sub-interval changes.
    *
     ---------- end definition ----------


Bands_Present                ( data      , discrete    )

  = ("1") +
    ("2") +
    ("3") +
    ("4") +
    ("5") +
```

```
    ("6") +
    ("7") +
    ("Pan").
    *
    The bands present for the sub-interval.
    *
     ---------- end definition ----------


BCH_Acct                    ( data      ,         )

  = Contact_Id +
    BCH_Data_Corrected_Error_Count +
    BCH_Pointer_Corrected_Error_Count +
    BCH_Data_Uncorrected_Error_Count +
    BCH_Pointer_Uncorrected_Error_Count.
    *
    Aggregate information on the number of Mission Data and
    Data Pointer Field BCH errors encountered in a data set,
    including counts of CADUs with BCH corrected errors and of
    CADUs with BCH uncorrected errors.
    *
     ---------- end definition ----------


BCH_Annotation              ( data      ,         )

  = Data_Field_Qual_Indicator +
    BCH_Bits_Corrected.
    *
    The BCH annotations for a CADU.
    *
     ---------- end definition ----------


BCH_Bits_Corrected          ( data      ,         )

  = Natural.
    *

    The total number of bits that were BCH corrected fort the
    mission data zone in the associated CADU.
    *
     ---------- end definition ----------


BCH_Chkd_VCDU               ( data      ,         )

  = Contact_Id +
    Sync +
    VCDU_Hdr_Bytes +
    [
     VCDU_Data |
     BCH_Corrected_Data
```

```
    ] +
    VCDU_Trailer +
    Sync_Annotation +
    CRC_Annotation +
    RS_Annotation +
    BCH_Annotation.
    *
    A VCDU that has completed the BCH Decode EDAC process which
    could have up to 3 bits corrected in the mission data field, along
    with the data quality indicators from frame sync, CRC, Reed_Solomon
    and BCH Decode process.
    *
     ---------- end definition ----------


BCH_Corrected_Data              ( data      ,         )

  = VCDU_Corrected_Mission_Data +
    272{Bit}272.
    *
    A VCDU with the mission data zone and/or pointer BCH corrected.
    *
     ---------- end definition ----------


BCH_Data_Corrected_Error_Count      ( data      ,         )

  = Natural.
    *
    Number of correctable BCH data field errors which occurred
    while CCSDS processing a raw wideband data set.
    *
     ---------- end definition ----------


BCH_Data_Uncorrected_Error_Count    ( data      ,         )

  = Natural.
    *
    Number of uncorrectable BCH data field errors which occurred
    while CCSDS processing a raw wideband data set.
    *
     ---------- end definition ----------


BCH_Failed_VCDU                 ( data      ,         )

  = Grade_3_Chkd_VCDU.

    *
    This is a VCDU which has failed the BCH decode algorithm.
    *

     ---------- end definition ----------
```

BCH_Pointer_Corrected_Error_Count (data ,)

= Natural.
*
Number of correctable BCH pointer field errors encountered
while CCSDS processing a raw wideband data set.
*
---------- end definition ----------

BCH_Pointer_Uncorrected_Error_Count (data ,)

= Natural.
*
Number of uncorrectable BCH pointer field errors encountered
while CCSDS processing a raw wideband data set.
*
---------- end definition ----------

BCH_Thres (data ,)

= Natural
*
The maximum number of BCH errors allowed before notifying operator.
*
---------- end definition ----------

BER (data ,)

= Real.
*
The approximate bit error rate computed from a channel's
contact period data (0.0...1.0) with TBD precision.
*
---------- end definition ----------

BER_Acct (data ,)

= Contact_Id +
BER.
*
The bit error rate accounting information.
*
---------- end definition ----------

BER_Thres (data ,)

= Real.

```
    *
    The maximum bit error rate allowed before notifying
    operator.
    *
     ---------- end definition ----------


Bit                         ( data      , primitive   )

  = [
     "0" |

     "1"
    ].
    *
    A binary digit.
    *
     ---------- end definition ----------


Boolean                     ( data      , primitive   )

  = [
     "TRUE" |
     "FALSE"
    ].
    *
    A flag containing the value of true or false.
    *
     ---------- end definition ----------


Browse_Acct                 ( data      ,         )

  = Sub_Intv_Id +
    Browse_File_Names.
    *
    Contains the names of all browse files associated with the
    subinterval.
    *
     ---------- end definition ----------


Browse_File                 ( store     ,         )

  = ([
    (Overlay_Wave + Mono_Band_Wave  ) |
    (Overlay_Wave + Multi_Band_Wave ) |
    (Overlay_Wave + Mono_Band_Wave  + Multi_Band_Wave)
    ]).
    *
    A datastore containing one or two reduced data-volume
    image files of the band files of a sub-interval.  If
    format 2, no browse file will exist.
```

```
    *
     ---------- end definition ----------


Browse_File_Names                ( data      ,          )

  = ([
     (Mono_Browse_File_Name) |
     (Multi_Browse_File_Name) |
     (Mono_Browse_File_Name + Multi_Browse_File_Name)
    ]).
    *
    The browse file name. Browse only exists if format 1 data.
    *
     ---------- end definition ----------


Browse_Store                     ( store     ,          )

  = {Aligned_Bands}.
    *
    A datastore that accumulates the Aligned_Bands data
    for Browse Processing until the sub-interval changes.
    *

     ---------- end definition ----------


Byte                             ( data      ,          )

  = 8{Bit}8
    *
    8 bits
    *.
     ---------- end definition ----------


CADU_Bit_Slip                    ( data      ,          )

  = *
    Indicates the bit slip total for the associated CADU.
    ### TBR ###
    *
     ---------- end definition ----------


CADU_Bit_Slip_Correction_Extent  ( data      ,          )

  = Natural.
    *
    The range of 0-3 bits.
    *
     ---------- end definition ----------
```

CADU_Bytes (data ,)

= 1036{Byte}1036.
*
A normal CADU extracted from the raw wideband data.
*
---------- end definition ----------

CADU_Check_Tolerance (data ,)

= Natural.
*
The range of CADUs from 0-3.
*
---------- end definition ----------

CADU_CRC_Error_Count (data ,)

= Natural.
*
Number of CRC errors encountered while CCSDS processing a Raw_WB_Data data set.
*
---------- end definition ----------

CADU_Flywheel_Count (data ,)

= Natural.
*
Number of CADUs with flywheel errors encountered while CCSDS processing a raw wideband data set.
*
---------- end definition ----------

CADU_Flywheel_Flag (data ,)

= Boolean.
*
A flag indicating that the associated CADU had flywheel errors.
*
---------- end definition ----------

CADU_Flywheel_Tolerance (data ,)

= Natural.
*

The range of CADUs from 0-3.
*
---------- end definition ----------

CADU_Missing_Count (data ,)

= Natural.
*
Number of missing CADUs noted during CCSDS processing of a raw wideband data set.
*
---------- end definition ----------

CADU_Polarity (data ,)

= *
The polarity for a particular CADU.
TBR
*
---------- end definition ----------

CADU_Polarity_Flag (data ,)

= Boolean.
*
A flag indicating the polarity of the associated CADU.
*
---------- end definition ----------

CADU_Rcv_Count (data ,)

= Natural.
*
Number of CADUs received in a raw wideband data set (Raw_WB_Data).
*
---------- end definition ----------

CADU_Search_Tolerance (data ,)

= Natural.
*
The range of CADUs from 1-3.
*
---------- end definition ----------

CADU_Sync_Error_Count (data ,)

= Natural.
*
Number of CADU sync errors encountered while CCSDS
processing a raw wideband data set (Raw_WB_Data).
*
---------- end definition ----------

CADU_Sync_Error_Flag (data ,)

= Boolean.
*
An indication of bit flips determined by the frame sync
process for the associated CADU
*
---------- end definition ----------

CADU_Sync_Lock_Error_Tolerance (data ,)

= Natural.
*
The range of 0-3 bits.
*
---------- end definition ----------

CADU_Sync_Marker_Check_Error_Tolerance (data ,)

= Natural.
*
The range of 0-3 bits.
*
---------- end definition ----------

Cal_Data (store ,)

= {Byte}.
*
The deinterleaved calibration data collected on a minor frame basis
for a given subinterval and is stored in the Cal_File.
*
---------- end definition ----------

Cal_Door_Activity_Status (data ,)

= Bit.
*
The status of the calibration door activity for a given period
of time. The Cal_Door_Activity_Status is located in "serial word P"
of the third PCD Major Frame.Minor Frame 83.Word 72. bits 2-3 of each

```
    PCD Cycle.
    *
     ---------- end definition ----------


Cal_File                    ( store     ,         )

  = {Byte}.
    *

    A file containing all of the calibration data received on a
    major frame basis for a given subinterval.
    *
     ---------- end definition ----------


Cal_File_Name               ( data      ,         )

  = File_Location +
    "L7_" +
    MF_Start_Time +
    "_" +
    Cal_File_Type +
    "_" +
    File_Version_Number.
    *
    The calibration file name.
    *
     ---------- end definition ----------


Cal_File_Type               ( data      , discrete    )

  = "Cal".
    *
    A string identifying the Cal file.
    *
     ---------- end definition ----------


Cal_Info                    ( store     ,         )

  = Cal_Door_Activity_Status.
    *
    Contains information relating the position of the
    calibration door for a given period of time.  The
    Cal_Info is generated on a subinterval basis.
    *
     ---------- end definition ----------


Cardinal                    ( data      , primitive   )

  = *
```

Any natural number excluding 0.
*.
---------- end definition ----------

CCA_Aggregate_Score (data ,)

= Real
*
Percentage of scene that is cloud covered
*.
---------- end definition ----------

CCA_Method (data ,)

= String.
*
An Operator defined indicator of the CCA method used:
Auto method 1 or Auto method 2.
*
---------- end definition ----------

CCA_Quadrant1_Score (data ,)

= Real
*
Percentage of first quadrant of a scene
that is covered with clouds.
*
---------- end definition ----------

CCA_Quadrant2_Score (data ,)

= Real
*
Percentage of second quadrant of a scene
that is covered with clouds.
*
---------- end definition ----------

CCA_Quadrant3_Score (data ,)

= Real
*
Percentage of third quadrant of a scene
that is covered with clouds.
*

---------- end definition ----------

```
CCA_Quadrant4_Score              ( data      ,          )

  = Real
    *
    Percentage of fourth quadrant of a scene
    that is covered with clouds.
    *
     ---------- end definition ----------


CCA_Ratio                        ( data      ,          )

  = Cardinal
    *
    Defines the reduction ratio to be used in the ACCA
    algorithm.  Valid values are: 4,8,16,32,48.
    *
     ---------- end definition ----------


Center_Latitude                  ( data      ,          )

  = Real

    *
    WRS scene center latitude. The angular distance, measured in degrees,
    north or south from the equator.
    *
     ---------- end definition ----------


Center_Longitude                 ( data      ,          )


  = Real.
    *
    WRS scene center longitude. Distance east or west on the earth's surface,
    measured as an arc of the equator between the meridian passing through
    a particular place and standard meridian.
    *
     ---------- end definition ----------


Chan_Acss_Acct                   ( data      ,          )

  = Contact_Id +
    Valid_CCSDS_Parms +
    CADU_Polarity +
    CADU_Bit_Slip +
    CADU_Sync_Error_Count +
    CADU_Rcv_Count +
    CADU_Flywheel_Count +
```

```
    CADU_Missing_Count.
    *
    Quality and accounting information derived from CCSDS
    processing on the channel access layer.
    *
     ---------- end definition ----------


Char                    ( data      , primitive   )

  = *
    1 byte character.
    *
     ---------- end definition ----------


Configuration_Items             ( data      ,         )

  = LGS_Channel_Id +
    LPS_Hardware_String_Id.
    *
    LPS configuration items.
    *

     ---------- end definition ----------


Contact_Ended                   ( data      ,         )

  = Bit.
    *
    A flag used to indicate the end of a contact period.
    *
     ---------- end definition ----------


Contact_File_Names              ( data      ,         )

  = {
     Sub_Intv_Id +
     Sub_Intv_File_Names
    }.
    *
    Contains all files associated with each subinterval of the contact.
    *
     ---------- end definition ----------


Contact_Id                      ( data      ,         )


  = LPS_Hardware_String_Id +
    LGS_Channel_Id +
    Contact_Start_Time +
```

```
    Contact_Stop_Time.
    *
    Identifies the data set for a contact period.
    *
     ---------- end definition ----------


Contact_Schedule_Id              ( data      ,          )

  = Natural.
    *
    A unique number assigned to a scheduled contact period or
    acquisition entry in Contact_Schedules store.
    *
     ---------- end definition ----------


Contact_Schedule_Start_Time      ( data      ,          )

  = Time.
    *
    Description:
    The scheduled contact start time.
    A time format either supported by local clock time or GMT time.
    *
     ---------- end definition ----------


Contact_Schedule_Stop_Time       ( data      ,          )

  = Time.
    *
    Description:
    The scheduled contact period stop time.
    A time format either local time or GMT time
    *
     ---------- end definition ----------


Contact_Schedules                ( store     ,          )

  = {
    @Contact_Schedule_Id +
    Contact_Schedule_Start_Time +
    Contact_Schedule_Stop_Time
    }.
    *
    A schedule containing multiple start/stop times that Landsat 7 spacecraft
    downlinks the wideband data to LGS.  The schedule is coming from LGS in
    a hardcopy form.
    *
     ---------- end definition ----------
```

Contact_Start_Time (store ,)

= Current_Time.
*
The contact period start time.
*
---------- end definition ----------

Contact_Stop_Time (data ,)

= Current_Time.
*
The contact period stop time.
*
---------- end definition ----------

CRC_Acct (data ,)

= Contact_Id +
CADU_CRC_Error_Count.
*
The cumulative sum generated in performing the CRC check.
*
---------- end definition ----------

CRC_Annotation (data ,)

= Boolean.
*
An indication as to whether the associated VCDU passed the CRC check.
*
---------- end definition ----------

CRC_Chkd_CADU (data ,)

= Contact_Id +
Sync +
VCDU_Bytes +
VCDU_Trailer +
Sync_Annotation +
CRC_Annotation.
*
A CADU that has completed the CRC checks.

NOTE:
The contact ID is not part of the annotation, but simply shown as information necessary to associate this CADU with a Contact_Id.

```
    *
     ---------- end definition ----------


CRC_Failed_CADU                ( data      ,          )

  = Ann_CADU.

    *
    This is a CADU which has failed the CRC Checksum.
    *
     ---------- end definition ----------


CRC_Thres                      ( data      ,          )

  = Natural.
    *
    The maximum number of CRC errors allowed before notifying operator
    *

     ---------- end definition ----------


Curr_VCID                      ( data      ,          )

  = VCID
    *
    The VCID of the current VCDU
    *
     ---------- end definition ----------


Current_Orbit                  ( store     ,          )

  = Orbit_Time +
    Orbit_Num.
    *
    The current orbit number and upper time range for the
    current orbit.
    *
     ---------- end definition ----------


Current_Sub_Intv_Id            ( store     ,          )

  = Natural.
    *
    The currently identified subinterval ID.
    *
     ---------- end definition ----------


Current_Time                   ( data      ,          )
```

```
  = Time.
    *
    The current system time.
    *
     ---------- end definition ----------


Cycle_Acct                    ( store     ,          )

  = 0{
     Num_PCD_Filled_MJF +
     Num_Avail_ADP +
     Num_Rejected_ADP +
     Num_Missing_ADP +
     Num_Avail_EDP +
     Num_Rejected_EDP +
     Num_Missing_EDP +
     1{
      Num_PCD_MNF_Sync_Errors +
      Num_PCD_Filled_MNF +
      Failed_PCD_Votes
     }128
    }4.
    *
    The buffer that contains the both minor and major
    frame accounting until the accounting for a complete
    PCD Cycle has been stored.
    *
     ---------- end definition ----------


DAN                           ( data      , primitive   )


  = *
    A Data Availability Notice (DAN) which is sent to the LPS DAAC to indicate
    the availability for transfer of the LPS output files associated with a
    particular contact period. It consists of information required by the
    LP-DAAC to transfer the files and then notify the LPS that files for
    that contact have been transferred.  Potential information included is
    the LPS Host Name, address, a contact Id and a list of file names.
    *
     ---------- end definition ----------


DAN_Suspended                 ( data      ,          )

  = Boolean.
    *
    This flag indicates whether DAN transfer is suspended.
    *
     ---------- end definition ----------
```

```
DAN_Transfer_State              ( store     , discrete   )

  = [
     "ENABLED" |
     "DISABLED"
    ].
    *
    This flag controls whether to send DANs to the LP DAAC or not.
    *
     ---------- end definition ----------


DAN_Transmission_Time           ( data      ,          )

  = Date +
    Time.
    *
    This object represents time a DAN was transmitted from LPS to LP DAAC.
    *
     ---------- end definition ----------


Data_Field_Qual_Indicator       ( data      , discrete   )

  = [
     "UNCORRECTABLE" |
     "CORRECTABLE" |
     "NO_ERRORS"
    ].
    *
    BCH checked data field quality for the associated CADU.
    *
     ---------- end definition ----------


Data_Missing_Message            ( data      ,          )

  = Message.
    *
    Error message indicating that PCD Data is missing due to
    missing VCDUs.
    *
     ---------- end definition ----------


Date                            ( data      , primitive  )


  = *
    This item will contain a year, month and day of the month
    *
     ---------- end definition ----------
```

```
Date_Of_Report_Data              ( data      ,          )

  = Date.

   *
   This object represents the date associated with the data in
   an LDTS file transfer summary report.
   *
    ---------- end definition ----------


Deinterleaved_Band_Data          ( data      ,          )

  = Format_Id +
   [
    Fmt1_Band_Data |
    Fmt2_Band_Data
   ].
   *
   All band data on a major frame basis that is deinterleaved
   according to band width.  Data is reversed, if necessary.
   *
    ---------- end definition ----------


Direction_Start                  ( data      ,          )

  = Address.
   *
   Indicates the beginning of a forward or reverse scan.
   *
    ---------- end definition ----------


Directive                        ( data      ,          )

  = [  RDC_Directive
     | RDP_Directive
     | MFP_Directive
     | IDP_Directive
     | PCD_Directive
     | LDT_Directive
     | MACS_Control_Drct
     | MACS_Modify_Config_Drct
     | MACS_Modify_Schedule_Drct
   ].
   *
   Inputs from LPS personnel to control LPS operation or change LPS system
   setup parameters.
   *


    ---------- end definition ----------
```

Drift_Rate (data ,)

= 1{Bit}32.

*
The SV Time Drift Characterization Data used, along with the Time of Last SV Clock Update, to correct the spacecraft time, reported in the PCD and video, for clock drift. The drift rate is located in the first PCD Major Frame.PCD Minor Frame 36-39.Word 72 of each Full_PCD_Cycle.
*
---------- end definition ----------

Drift_Time (data ,)

= Sub_Intv_Id +
Drift_Rate
*
A structure that contains the Sub_Intv identification and the Drift_Rate used to calculate the difference between the spacecraft time and the actual time of each ETM plus Major Frame on a Sub_Intv basis.
*
---------- end definition ----------

DTA (data , primitive)

= *
An unsolicited message from the LP DAAC indicating the status of the LP DAAC's attempt to transfer a set of LPS output files associated with a given contact ID.
*
---------- end definition ----------

DTA_Time_Of_Receipt (data ,)

= Date +
Time.

*
This object represents the time that a DTA from the LP DAAC for a contact ID was received by LPS.
*
---------- end definition ----------

End_Of_Contact_Flag (data ,)

= Boolean.

*
Indicates whether the end of the contact period has been reached
*
---------- end definition ----------

Ending_Row (data ,)

= WRS_Row_Nominal.
*
The ending row of the last scene in a sub-interval. The information is determined by PCDS and is given to MACS for inclusion of metadata file via PCD_Acct.
*
---------- end definition ----------

Eol_Thr (data ,)

= Natural.
*
The threshold value for the number of missing end of line code minor frames for a subinterval.
*
---------- end definition ----------

EPA (data ,)

= Real.
*
Attitude estimate represented as an Euler Parameter.
*
---------- end definition ----------

EPA1 (data ,)

= EPA.
*
First EPA of the quaternion defining rotation.
*
---------- end definition ----------

EPA2 (data ,)

= EPA.
*
The second EPA of the quaternion defining the rotation.
*
---------- end definition ----------

EPA3 (data ,)

= EPA.
*
The third EPA of the quaternion defining the rotation.
*
---------- end definition ----------

EPA4 (data ,)

= EPA.
*
The fourth EPA of the quaternion defining the rotation.
*
---------- end definition ----------

Ephem_Position_Lower (data ,)

= Real.
*
The smallest valid ephemeris position data point which is
-8.3886x10^6 meters.
*
---------- end definition ----------

Ephem_Position_Upper (data ,)

= Real.

*
The largest valid ephemeris position data point which is
+8.3886x10^6 meters.
*
---------- end definition ----------

Ephem_Position_x (data ,)

= Real.
*
The ephemeris/position x-coordinate value.
*
---------- end definition ----------

Ephem_Position_y (data ,)

= Real.
*
The ephemeris/position y-coordinate value.
*

---------- end definition ----------

Ephem_Position_z (data ,)

= Real.
*
The ephemeris/position z-coordinate value.
*
---------- end definition ----------

Ephem_Velocity_Lower (data ,)

= Real.
*
The smallest valid ephemeris data point value which is
-8.0 meters./ms
*
---------- end definition ----------

Ephem_Velocity_Upper (data ,)

= Real.
*
The largest valid ephemeris data point value which is
+8.0 meters/ms.
*
---------- end definition ----------

Ephem_Velocity_x (data ,)

= Real.
*
The velocity x-coordinate.
*

---------- end definition ----------

Ephem_Velocity_y (data ,)

= Real .
*
The velocity y-coordinate value.
*

---------- end definition ----------

```
Ephem_Velocity_z                ( data      ,         )

  = Real.
   *
   The velocity z-coordinate value.
   *
    ---------- end definition ----------


Ephemeris                       ( data      ,         )

  = Ephem_Velocity_x
   + Ephem_Velocity_y
   + Ephem_Velocity_z
   + Ephem_Position_x
   + Ephem_Position_y
   + Ephem_Position_z.
   *
   The position and velocity components of spacecraft at a
   specific time.  The ephemeris data is located in
   PCD Major Frame 0 and 2.PCD Minor Frame 50-73.Word 72 or
   PCD Major Frame 1 and 3.PCD Minor Frame 16-39.Word 72.
   *
    ---------- end definition ----------


Est_Time                        ( data      ,         )

  = Real.
   *
   The estimated major frame time in 1/16th of a millisecond.
   *
    ---------- end definition ----------


Est_Used_Flag                   ( data      ,         )

  = Boolean.
   *
   A true flag indicating that the actual major frame time was in
   error.  The estimated time should be used.
   *
    ---------- end definition ----------


ETM_Data_Format                 ( data      ,         )

  = Natural.
   *
   The ETM+ data format type provided by MFPS and is included
   in the metadata file.
   *
    ---------- end definition ----------
```

ETM_Plus_LOS_x (data ,)

= Real.
*
The x-coordinate of the line of sight vector.
*
---------- end definition ----------

ETM_Plus_LOS_y (data ,)

= Real.
*
The y-coordinate for the line of sight vector.
*
---------- end definition ----------

ETM_Plus_LOS_z (data ,)

= Real.
*
The z-coordinate for the line of sight vector.
*
---------- end definition ----------

ETM_Plus_To_Body_Trans (data ,)

= String.
*
Parameters used to transform the longitude, latitude, sun elevation,
and sun azimuth from ETM Plus to Spacecraft Body.
*
---------- end definition ----------

Exp_Eol_Ptr (data ,)

= Ann_VCDU.
*
The pointer to an annotated VCDU in which the end of line
code minor frames are expected to be.
*
---------- end definition ----------

Exp_Mnf_Ctr (data ,)

= Natural
*
The expected minor frame counter used for minor frame counter verification. *.

```
    ---------- end definition ----------


Exp_VCDU_Ctr                ( data      ,         )

  = Natural.
   *
   The expected VCDU counter used for VCDU sequence counter
   verification.
   *
    ---------- end definition ----------


Faild_CADUs                 ( store     ,         )

  = {

    @Contact_Id +
    Sync +
    [
     PN_Decoded_CADU_Bytes |
     [
      VCDU_Bytes |
      VCDU_Hdr_Bytes + VCDU_Data
     ] +
     VCDU_Trailer
    ] +
    Sync_Annotation +
    (CRC_Annotation) +
    (RS_Annotation)
   }.
   *
   The set of Grade 3 Failed CADUs and BCH Failed VCDUs for
   each data set processed by the RDPS and stored in such a
   way as to allow retrieval on a contact period basis.
   *
    ---------- end definition ----------


Failed_Mjf_Data             ( store     ,         )

  = {
    @Contact_Id +
    Major_Frame_VCDU_Set
   }.
   *
   A Major_Frame_VCDU_Set that is missing either a major frame synchronization
   or both EOL code minor frames. This store is identified on
   a contact period basis.
   *
    ---------- end definition ----------


Failed_PCD_Votes            ( store     ,         )
```

= Natural.
*
The number of failed attempts to perform a successful majority rule of three consecutive PCD information words (PCD_Info_Word). The majority vote is performed using the three consecutive words that follow the sync byte of each PCD data word cycle.
*
---------- end definition ----------

Failed_Votes (store ,)

= Natural.
*
A buffer used to accumulate the number of unsuccessful attempts to perform a majority vote. The count is based on each PCD Minor Frame within a sub-interval.
*
---------- end definition ----------

Failed_Votes_Message (data ,)

= Message.
*
Error message indicating the number of failed attempts to perform a majority vote to determine the PCD_Info_Word.

*
---------- end definition ----------

FHS_Err (data ,)

= 12{Char}12.
*
The first half scan time error.
*
---------- end definition ----------

File_Location (data ,)

= String.
*
Location of a file.
*
---------- end definition ----------

File_Version_Number (data ,)

= Natural.
*
Description:
A unique number assigned to the version of metadata file associated with a specific sub-interval, contact period.
*
---------- end definition ----------

Fill_SCID (data ,)

= 6{Bit}6.
*
A SCID that contains all ones indicating a fill CADU.
*
---------- end definition ----------

Fill_Value (data ,)

= Byte
*
Predefined fill value. It is an operator input.
*.
---------- end definition ----------

First_PCD_MJF_Time (data ,)

= Time.
*
The Major_Frame_Time that corresponds to the spacecraft time of the first PCD Major Frame of a sub interval.
*
---------- end definition ----------

Flush_Minor_Frame (data ,)

= Contact_Ended.
*
A flag used to indicate that all partially filled PCD Minor Frames

should be completed using a predefined fill pattern due to the end of a contact period.
*
---------- end definition ----------

Fmt (data ,)

= Natural.
*
A value indicating whether the Mono, Multi1, Multi2 and

```
    Multi3 band parameters are to be used for format 1 or 2
    scene data:  Format 1 = "0", Format 2 = "1"
    *
     ---------- end definition ----------


Fmt1_Align_Data                ( data      ,         )

  = 0{
     Fill_Value
    }Max_Alignment_Value +
    Fmt1_Band_Data +
    0{
     Fill_Value
    }Max_Alignment_Value.
    *
    Format 1 aligned data.  It includes bands 1 through 6.
    *
     ---------- end definition ----------


Fmt1_Band_Data                 ( data      ,         )

  = 5{16{Band_Det_Data}16}5 +
    4{Band6_Det_Data}4.
    *
    Format 1 deinterleaved detector data.  It includes bands 1
    through 5 and band 6.
    *
     ---------- end definition ----------


Fmt2_Align_Data                ( data      ,         )

  = 0{
     Fill_Value
    }Max_Alignment_Value +
    Fmt2_Band_Data +
    0{
     Fill_Value
    }Max_Alignment_Value.
    *
    Format 2 aligned data.  It includes bands 6, 7, and Pan.
    The total sum of the fill values will equal the maximum
    alignment value.
    *
     ---------- end definition ----------


Fmt2_Band_Data                 ( data      ,         )

  = 4{Band6_Det_Data}4 +
    16{Band_Det_Data}16 +
    2{32{Pan_Det_Data}32}2.
```

```
    *
    Format 2 deinterleaved detector data.  It includes band 6,
    band 7, and Pan.
    *
     ---------- end definition ----------


Fmt_Status                  ( data      ,         )

  = Message

    *
    Status indicating the success or failure of the validation
    of Band_Parms.Fmt.
    *
     ---------- end definition ----------


Format_Id                   ( data      ,         )

  = Bit.
    *
    Format 1/2 ID. Word 7 bit 4 of PCD/Status Data.
    Format 1 = "0", Format 2 = "1".
    *
     ---------- end definition ----------


Full_Mjf_Thr                ( data      ,         )

  = Natural.
    *
    The threshold value for the number of full missing major frames
    occurring during a subinterval.
    *
     ---------- end definition ----------


Full_PCD_Cycle              ( store     ,         )

  = Sub_Intv_Id +
    4{PCD_Major_Frame}4.
    *
    Consists of four PCD Major Frames whose Major Frame Id is sequentially
    numbered from 0-3.  The full PCD cycle refers to a complete set of a
    PCD table of data.
    *
     ---------- end definition ----------


Gain_Change_Flag            ( data      , primitive   )

  = [
```

```
    "0" |
    "1"
   ].
   *
   A bit indicator of word 8 in the PCD/Status field.
   Low gain = "0". High gain = "1"
   *
    ---------- end definition ----------


Gap_Tagged_VCDUs              ( data      ,         )

  = Mjf_CADU_Rcvd_Cnt{
    Ann_VCDU +

    Num_Missing_VCDUs
   }Mjf_CADU_Rcvd_Cnt.
   *
   Contains an annotated VCDU and the number of missing VCDUs.
   *
    ---------- end definition ----------


Grade_3_Chkd_VCDU             ( data      ,         )

  = Contact_Id +
    Sync +
    VCDU_Hdr_Bytes +
    VCDU_Data +
    VCDU_Trailer +
    Sync_Annotation +
    CRC_Annotation +
    RS_Annotation.
   *
   A CADU that has completed the CCSDS Grade 3 processing and
   the data quality annotation from frame sync, CRC, and
   Reed-Solomon EDAC processes.

   NOTE:
   The contact ID is not part of the annotation, but simply
   shown as information necessary to associate this VCDU with
   a Contact_Id.
   *
    ---------- end definition ----------


Gyro_Data                     ( data      ,         )

  = Byte.
   *
   One byte of a 24 bit value, in two's complement with the most
   significant byte first, for each gyro axis in the PCD.
   The result of three orthogonal axes of the inertial
   measurement unit(IMU) which are sampled by the spacecraft
```

```
    every 64 milliseconds.
    *
     ---------- end definition ----------


Horizontal_Display_Shift          ( data      ,          )

  = Real.
    *
    The horizontal display shift of WRS scene. The information
    is determined by PCDS and is given to MACS for inclusion of
    metadata file via PCD_Acct.
    *
     ---------- end definition ----------


IDP_ACCA_Status                   ( data      ,          )

  = Message.
    *
    This status reflects whether ACCA failed or succeeded.  If
    ACCA failed, what type of failure
    *
     ---------- end definition ----------


IDP_Acct                          ( store     ,          )


  = {
    @Sub_Intv_Id +
    Browse_File_Names +
    Bands_Present +
    Band_File_Names +
    PCD_Scene_Count +
    0{
     Band_Gains +
     Gain_Change_Flag +
     (
      CCA_Method +
      ACCA
     )
    }PCD_Scene_Count
    }.
    *
    Information such as filenames, band numbers, sub-interval
    identifiers and sub-interval start and stop times for
    Browse, Band and ACCA tasks. There is multiple CCA information,
    one for each full scene if the format id is 1. There is no
    ACCA for format 2 data.
    *
     ---------- end definition ----------
```

```
IDP_Band_Parms                ( data      ,         )

  = (Mono) +
    (Multi1) +
    (Multi2) +
    (Multi3) +
    (Fmt) +
    (Subs) +
    (Wave) +
    (CCA_Method) +
    (CCA_Ratio)
    *
    Operator-defined parameters specifying which bands to
    process for ACCA and Browse (Mono, Multi1, Multi2, Multi3),
    whether the band parameters are to be used for format 1
    or 2 scene data (Fmt), and the reduction ratio to use
    when reducing browse image data by subsampling and by
    wavelets (Subs and Wave). CCA_Method and CCA_Ratio are
    two user-defined ACCA parameters.
    *

     ---------- end definition ----------


IDP_Directive                 ( data      ,         )

  = IDP_Band_Parms.
    *
    The directives and parameters sent to the IDPS from the MACS.
    *
     ---------- end definition ----------


IDP_Setup_Status              ( data      ,         )

  = [
     Mono_Status |
     Multi1_Status |
     Multi2_Status |
     Multi3_Status |

     Fmt_Status |
     Subs_Status |
     Wave_Status
    ].
    *
    Status indicating the success or failure of the validation
    on monochrome and multi band parameters, the format ID, and
    the subsampling and wave ratios.
    *
     ---------- end definition ----------


IDP_Status                    ( data      ,         )
```

```
  = [
     IDP_Setup_Status |
     IDP_ACCA_Status
    ].
    *
    Status messages sent from IDPS to the MACS
    *
     ---------- end definition ----------


Info_Word                    ( data      ,         )

  = [
     ADS |
     Gyro_Data |
     Sync_Word |
     Major_Frame_Id |
     Subcomm_Word
    ].
    *
    An eight-bit word that contains the mission-related
    telemetry describing the attitude, ephemeris, jitter, and
    other data that describes the status of the spacecraft and
    its subsystems.
    *

     ---------- end definition ----------


Info_Word_1                  ( data      ,         )

  = Info_Word.
    *
    The first PCD information word that follows the PCD information
    word data cycle sync pattern.
    *
     ---------- end definition ----------


Info_Word_2                  ( data      ,         )

  = Info_Word.
    *
    The second information word that follows the PCD information
    word data cycle sync pattern.
    *
     ---------- end definition ----------


Info_Word_3                  ( data      ,         )

  = Info_Word.
```

```
    *
    The third PCD information word that follows the PCD information
    word data cycle sync pattern.
    *
     ---------- end definition ----------


Info_Word_Missing              ( data      ,          )

  = Natural.
    *
    A value that indicates how many PCD data words are missing.
    *
     ---------- end definition ----------


Instrument_Id                  ( data      ,          )

  = Natural.
    *
    The ETM+ instrument ID.
    *

     ---------- end definition ----------


Integer                        ( data      , primitive )

  = *
    An integer
    *
     ---------- end definition ----------


Internal_DAN                   ( data      ,          )

  = Contact_Id +
    Contact_File_Names.
    *
    This object contains all of the contact specific information required to generate
    a Data Availability Notice (DAN).
    *
     ---------- end definition ----------


Inverted_CADU_Bytes            ( data      ,          )

  = 1036{Byte}1036
    *
    An inverted CADU extracted from the raw wideband data.
    *
     ---------- end definition ----------
```

```
Inverted_Sync                ( data      ,          )

  = 4{Byte}4
   *
   Inverted Hex '1ACFFC1D'
   *.
    ---------- end definition ----------


Lat_And_Long                 ( store     ,          )

  = Sub_Intv_Id +

   Latitude +
   Longitude +
   Major_Frame_Time +
   Cal_Door_Activity_Status.
   *
   The calculated latitude and longitude for each PCD Major Frame.
   *
    ---------- end definition ----------


Latitude                     ( data      ,          )

  = Real.
   *
   The angular distance, measured in degrees, north or south
   from the equator..
   *
    ---------- end definition ----------


LDT_DAN_Status               ( data      ,          )

  = Message.

   *
   This flow will contain either an indication that a DAN
   was successfully sent (milestone) or that problems
   were detected while attempting to transmit a DAN.
   *
    ---------- end definition ----------


LDT_Delete_Files_Drct        ( data      ,          )

  = Contact_Id
   *
   DESCRIPTION:
   This control directive specifies a contact for which all
   LPS Output files are to be deleted.
   *
    ---------- end definition ----------
```

```
LDT_Delete_Files_Status          ( data      ,          )

  = Message.
    *
    This is the contact ID associated with a set of files
    which have been deleted.
    *
     ---------- end definition ----------


LDT_Directive                    ( data      ,          )

  = [ LDT_Resend_DAN_Drct
    | LDT_Retain_Files_Drct
    | LDT_Rpt_File_Xfer_Sum_Drct
    | LDT_Delete_Files_Drct
    | LDT_Enable_File_Xfer_Drct
    | LDT_Disable_File_Xfer_Drct
    ]
    *
    DESCRIPTION:
    The control directives from LPS personnel to control the LPS data transfer
    to LP DAAC.

    *
     ---------- end definition ----------


LDT_Disable_File_Xfer_Drct       ( data      , primitive   )

  = *
    A directive sent to LDTS for disabling the transfer of DAN to
    LP DAAC.
    *

     ---------- end definition ----------


LDT_DTA_Status                   ( data      ,          )

  = Message.
    *
    This is a message which contains the contact ID associated with a set of LPS output
    files for which LPS has received a DTA from the LP DAAC.  Also included is relevant
    transfer status information (e.g., success, failure, reason(s), etc.).
    *
     ---------- end definition ----------


LDT_Enable_File_Xfer_Drct        ( data      , primitive   )

  =
```

```
    *
    A directive sent to LDTS for enabling the transfer of DAN to
    LP DAAC.
    *

     ---------- end definition ----------


LDT_Output_File_Info              ( store      ,          )

  = @Contact_Id +
    DAN_Suspended +
    Marked_For_Retention +
    Time_Available +
    Time_Deleted +
   {DAN_Transmission_Time } +
   {DTA_Time_Of_Receipt } +
    Contact_File_Names.

    *
    This data store contains all state information about LPS output files that is of
    concern to LDTS.
    *
     ---------- end definition ----------


LDT_Resend_DAN_Drct               ( data       ,          )

  = Contact_Id

    *
    This directive is used to manually trigger LDTS to resend
    a DAN to the LP DAAC.
    *
     ---------- end definition ----------


LDT_Retain_Files_Drct             ( data       ,          )


  = Contact_Id

    *
    This directive is used to retain the LPS files no matter what the file is
    successfully transferred to LP DAAC or not. The retained files will reside
    on the LPS indefinitely unless the operator deletes them manually using
    'LPS_File_Deletion' directive. A contact schedule list menu will be
    prompted to the operator to choose which contact schedule files should be
    retained.
    *
     ---------- end definition ----------


LDT_Retain_Files_Status           ( data       ,          )
```

```
  = Message.
   *
   This represents is a status message indicating the success or failure of
   the attempt to retain all files associated with a particular contact ID.
   *
    ---------- end definition ----------


LDT_Rpt_File_Xfer_Sum_Drct          ( data      ,          )

  = Date
   *
   DESCRIPTION:
   This directive specifies a date for which a File Transfer
   Summary Report is to be generated.
   *
    ---------- end definition ----------


LDT_Send_DAN                        ( data      ,          )

  = Contact_Id +
   Contact_File_Names.
   *
   Indicates that all files for the contact have been generated and provides
   a list of completed LPS output filenames..
   *
    ---------- end definition ----------


LDT_Status                          ( data      ,          )

  = [ LDT_DAN_Status |
     LDT_DTA_Status |
     LDT_Retain_Files_Status |
     LDT_Delete_Files_Status
   ]
   *
   Status messages returned from LDTS
   *.
    ---------- end definition ----------


LGS_Channel_Id                      ( data      ,          )

  = String.
   *
   The identifier for an LGS connection used by an LPS string.
   *
    ---------- end definition ----------


Longitude                           ( data      ,          )
```

= Real.
*
Distance east or west on the earth's surface, measured as an arc of the equator between the meridian passing through a particular place and standard meridian.
*
---------- end definition ----------

Lower_Left_Corner_Latitude (data ,)

= Real

*
WRS scene lower left corner latitude. The angular distance, measured in degrees, north or south from the equator.
*
---------- end definition ----------

Lower_Left_Corner_Longitude (data ,)

= Real

*
WRS scene lower left corner longitude. Distance east or west on the earth's surface, measured as an arc of the equator between the meridian passing through a particular place and standard meridian.
*
---------- end definition ----------

Lower_Right_Corner_Latitude (data ,)

= Real
*
WRS lower right corner latitude. The angular distance, measured in degrees, north or south from the equator.
*
---------- end definition ----------

Lower_Right_Corner_Longitude (data ,)

= Real
*
WRS lower right corner longitude. Distance east or west on the earth's surface, measured as an arc of the equator between the meridian passing through a particular place and standard meridian.
*
---------- end definition ----------

```
LPS_Acct                    ( store     ,         )

  = RDC_Acct +
    RDP_Acct +
    MFP_Acct +
    PCD_Acct +
    IDP_Acct.
    *
    The LPS quality and accounting information created by LPS subsystems. The

    information is used to construct the Metadata_File on a subinterval basis.
    The quality and accounting files are retrieved by the MACS once MACS
    receives the Sub_Intv information from the MFPS.
    *
     ---------- end definition ----------


LPS_Configuration           ( store     ,         )

  = LPS_Hardware_String_Id +
    LGS_Channel_Id +
    LPS_Software_Version_Number +
    Spacecraft_Id +
    Instrument_Id +
    File_Version_Number.
    *
    The setup information used to startup LPS.
    *

     ---------- end definition ----------


LPS_File_Transfer_Status    ( data      , primitive   )

  =
    *
    This string will indicate the success or failure of an attempt by the
    LP DAAC to transfer LPS output files from LPS to the LP DAAC.
    *
     ---------- end definition ----------


LPS_Files                   ( data      ,         )

  = PCD_File
    + Band_File
    + Browse_File
    + Cal_File
    + MSCD_File
    + Metadata_File
    *
    This includes all L0R files generated by LPS.
    *.
```

---------- end definition ----------

LPS_Hardware_String_Id (data ,)

= Natural.
*
Description:
The data string number configured by LGS/LPS personnel. It is configured in the LPS_Configuration store.
*
---------- end definition ----------

LPS_Journal (store ,)

= {
[
RDC_Status |
RDP_Status |
MFP_Status |
PCD_Status |
IDP_Status |
MACS_Directive_Dispatch_Status |

MACS_Metadata_Generation_Status |
MACS_Control_Status |
MACS_Modify_Schedule_Status |
MACS_Modify_Config_Status |
LDT_Status
]
}.
*
A system log file containing all activities which occurred
during LPS operation
*

---------- end definition ----------

LPS_Report (data ,)

= [Report_RDC_Data_Capture_Sum
| Report_RDP_Return_Link_QA
| Report_MFP_L0R_QA
| Report_LDT_File_Xfer_Sum
].
*
The processing summary information generated by the RDCS, RDPS, and LDTS, respectively. These summary information serves as LPS processing monitoring media by LPS personnel. The information is sent by the subsystem by either contact period basis, or by daily basis.
*
---------- end definition ----------

```
LPS_Software_Version_Number          ( data      ,          )

  = Natural.
    *
    A unique number assigned to specific LPS software version.
    *
     ---------- end definition ----------


LPS_Status                           ( data      ,          )

  = [
     MACS_Directive_Dispatch_Status |
     MACS_Metadata_Generation_Status |
     MACS_Control_Status |
     MACS_Modify_Config_Status |
     MACS_Modify_Schedule_Status |
     RDC_Status |
     RDP_Status |
     MFP_Status |
     PCD_Status |
     IDP_Status |
     LDT_Status
    ].
    *
    The status information used to monitor LPS processing.
    *
     ---------- end definition ----------


MACS_Control_Drct                    ( data      , discrete    )

  = [ "Startup"
    | "Shutdown"
    ]

    *
    A method of controlling LPS system activities.
    *
     ---------- end definition ----------


MACS_Control_Status                  ( data      ,          )

  = Message.
    *
    An error message during processing LPS system control directives.
    *

     ---------- end definition ----------
```

```
MACS_Directive_Dispatch_Status      ( data      , discrete    )

  = [
     "SUCCESS" |
     "FAIL"
    ].
    *
    The directive dispatching status.
    *
     ---------- end definition ----------


MACS_Metadata_Generation_Status     ( data      ,          )

  = Message.

    *
    Messages indicating the results of the metadata file generation process.
    *
     ---------- end definition ----------


MACS_Modify_Config_Drct             ( data      ,          )

  = (LPS_Hardware_String_Id) +
    (LGS_Channel_Id) +
    (LPS_Software_Version_Number) +
    (Spacecraft_Id) +
    (Instrument_Id) +
    (File_Version_Number).
    *
    A LPS configuration modification directive keyed by LPS personnel to
    replace the existing LPS_Configuration.
    *
     ---------- end definition ----------


MACS_Modify_Config_Status           ( data      ,          )

  = Message.
    *
    A text string containing error information.
    *
     ---------- end definition ----------


MACS_Modify_Schedule_Drct           ( data      ,          )

  = (Contact_Schedule_Id) +
    Modification_Type +

    Contact_Schedule_Start_Time +
    Contact_Schedule_Stop_Time.
    *
```

A contact schedule modification directive to update an existing contact schedule, to insert a new contact schedule or to delete an existing contact schedule.
*
---------- end definition ----------

MACS_Modify_Schedule_Status (data ,)

= Message.
*
A text string containing error information.
*

---------- end definition ----------

Maj_Vote_Tol (data ,)

= Natural
*
A value indicating the minimum number of identical bits that are required for the data word group majority vote.
*.
---------- end definition ----------

Major_Frame_Acct (data ,)

= Num_PCD_Filled_MJF +
Num_Avail_ADP +
Num_Rejected_ADP +
Num_Missing_ADP +
Num_Avail_EDP +
Num_Rejected_EDP +
Num_Missing_EDP.
*
PCD Major Frame related statistics.
*
---------- end definition ----------

Major_Frame_Id (data ,)

= Byte.
*
Unique identifier, from 0 to 3, which indicates the position of the PCD Major Frame within the PCD Cycle.
*
---------- end definition ----------

Major_Frame_Time (data ,)

= Actual_Time +
(Est_Time) +
Est_Used_Flag.
*
The time associated with one major frame. This time is
extracted from minor frames two through seven. It is
calculated according to the minor frame format as stated in
the DFCB.
*

---------- end definition ----------

Major_Frame_VCDU_Set (data ,)

= Gap_Tagged_VCDUs +
Exp_Eol_Ptr +
Mjf_CADU_Seq_Err_Cnt +
Mjf_CADU_Fly_Cnt +
Mjf_CADU_Rcvd_Cnt +
Mnf_Ctr_Err.
*
Structure with a pointer to a set of VCDUs for one scan bit value, that is, one major frame's worth of data.
*
---------- end definition ----------

Majority_Vote_Failure (data ,)

= Bit.
*
A flag indicating a failure or a success when performing
a majority vote to determine the PCD_Info_Word.
*
---------- end definition ----------

Marked_For_Retention (data , discrete)

= [
"RETAIN" |
"DELETE"
].
*
A flag indicating whether LPS will automatically delete all files
associated with a contact.
*
---------- end definition ----------

Max_Alignment_Value (data ,)

= Natural.

```
    *
    The maximum value allowable as an alignment value.
    *
     ---------- end definition ----------


Message                    ( data      , primitive   )

  = *
    The messages used to monitor LPS processing. This will
    contain information such as the message ID, a time tag, and
    optional string messages
    *
     ---------- end definition ----------


Metadata_File              ( store     ,          )

  = Metadata_Header +
    Mjf_CADU_BER_Cnt +
    Mjf_CADU_Sync_Err_Cnt +
    Mjf_CADU_Rcvd_Cnt +
    Mjf_CADU_Missing_Cnt +

    VCDU_Hdr_Err_Count +
    Mjf_CADU_BCH_Corr_Cnt +
    Mjf_CADU_BCH_Uncorr_Cnt +
    Mjf_CADU_BCH_Bits_Corr +
    Mjf_Full_Fill_Cnt +
    Mjf_Part_Fill_Cnt +
    Mjf_Time_Code_Err_Cnt +
    Failed_PCD_Votes +
    Num_PCD_MNF_Sync_Errors +
    Num_PCD_Filled_MNF +
    Num_PCD_Filled_MJF +
    Num_Avail_ADP +
    Num_Rejected_ADP +
    Num_Missing_ADP +
    Num_Avail_EDP +
    Num_Rejected_EDP +
    Num_Missing_EDP +
    0{
     Sub_Intv_Scene_Num +
     WRS_Path_Nominal +
     WRS_Row_Nominal +
     Scene_Center_Time +
     Scene_Center_Scan_Num +
     Horizontal_Display_Shift +
     Center_Latitude +
     Center_Longitude +
     Upper_Left_Corner_Latitude +
     Upper_Left_Corner_Longitude +
     Upper_Right_Corner_Latitude +
     Upper_Right_Corner_Longitude +
```

```
    Lower_Left_Corner_Latitude +
    Lower_Left_Corner_Longitude +
    Lower_Right_Corner_Latitude +
    Lower_Right_Corner_Longitude +
    Sun_Azimuth +
    Sun_Elevation +
    CCA_Quadrant1_Score +
    CCA_Quadrant2_Score +
    CCA_Quadrant3_Score +
    CCA_Quadrant4_Score +
    CCA_Aggregate_Score +
    Gain_Change_Flag +
    Band_Gains
   }PCD_Scene_Count.
   *
   A file containing information on the Level OR processing
   for a specific subinterval.
   *
    ---------- end definition ----------


Metadata_File_Name              ( data      ,         )

  = File_Location +
   "L7_" +
   MF_Start_Time +
   "_" +
   Metadata_File_Type +
   "_" +
   File_Version_Number.
   *
   The metadata file name.
   *
    ---------- end definition ----------


Metadata_File_Type              ( data      , discrete    )


  = "Metadata".
   *
   A string identifying the Metadata file.
   *
    ---------- end definition ----------


Metadata_Header                 ( data      ,         )

  = Current_Time +
   File_Version_Number +
   LPS_Hardware_String_Id +
   LPS_Software_Version_Number +
   Spacecraft_Id +
   ETM_Data_Format +
```

```
    Instrument_Id +
    Contact_Start_Time +
    Orbit_Num +
    MF_Start_Time +
    MF_Stop_Time +
    Mjf_Count +
    Num_PCD_MJF +
    First_PCD_MJF_Time +
    WRS_Path_Nominal +
    Starting_Row +
    Ending_Row +
    PCD_File_Name +
    Browse_File_Names +
    Cal_File_Name +
    MSCD_File_Name +
    Band_File_Names +
    Bands_Present +
    ETM_Data_Format.
    *
    The overall accounting information associated with a subinterval data.
    Browse_File_Names are only for data format type 1.
    *
     ---------- end definition ----------


MF_Start_Time                ( data      ,          )

  = Major_Frame_Time.
    *
    Subinterval start time.
    *
     ---------- end definition ----------


MF_Stop_Time                 ( data      ,          )

  = Major_Frame_Time.
    *
    Subinterval stop time.
    *
     ---------- end definition ----------


MFP_Acct                     ( store     ,          )

  = {
    @Sub_Intv_Id +
    Mjf_VCDU_QA +
    Cal_File_Name +

    MSCD_File_Name
   }.
   *
   All quality and accounting information that is received on
```

```
    a major frame basis for a given subinterval. It consists of
    information on the major frame processing data, the
    Calibration filename and MSCD filename.
    *
     ---------- end definition ----------


MFP_Cal_Status                ( data      ,          )

  = Message
    *
    Status messages as a result of processing the
    calibration data
    *
     ---------- end definition ----------


MFP_Directive                 ( data      ,          )

  = [
     MFP_Rpt_L0R_QA_Drct |
     MFP_Parms |
     MFP_Thresholds
    ].
    *
    Description:
    The directive sent to MFPS to control the MFPS subsystem .
    *
     ---------- end definition ----------


MFP_Mjf_Status                ( data      ,          )

  = Message
    *
    Threshold error messages from MFPS.
    *.
     ---------- end definition ----------


MFP_MSCD_Status               ( data      ,          )

  = Message.
    *
    This is the status messages as a result of MSCD data extraction.
    Any resulting error or status messages will be sent to the MACS.
    *
     ---------- end definition ----------


MFP_Parms                     ( store     ,          )

  = (Sensor_Alignment_Info) +
    (Fill_Value) +
```

```
    (Sub_Intv_Delta) +
    (Mjf_Data_Rate) +
    (Max_Alignment_Value) +
    (Time_Range_Tol) +
    (Part_Mnf_Tol) +
    (Maj_Vote_Tol).
    *
    The MFPS setup parameters.

    *
     ---------- end definition ----------


MFP_Rpt_L0R_QA_Drct              ( data      ,          )

  = Sub_Intv_Id.
    *
    Upon receipt of this control directive from LPS personnel, the MFPS will
    gather the Level 0R quality and accounting information for
    the specified sub-interval and forward the information to the
    MACS for either display or hardcopy report.
    *
     ---------- end definition ----------


MFP_Setup_Status                 ( data      ,          )

  = Message
    *
    A return status to indicate acceptance or rejection of operator controls.
    *.
     ---------- end definition ----------


MFP_Status                       ( data      ,          )

  = [
     MFP_Setup_Status |
     MFP_Mjf_Status |
     MFP_Cal_Status |
     MFP_MSCD_Status
    ].
    *
    Status messages sent to MACS from the MFPS.
    *
     ---------- end definition ----------


MFP_Thresholds                   ( store     ,          )

  = (Mjf_CADU_Seq_Err_Thr) +
    (Scan_Dir_Thr) +
    (Sync_Thr) +
    (Mnf_Ctr_Thr) +
```

```
    (Eol_Thr) +
    (Tc_Thr) +
    (Full_Mjf_Thr) +
    (Part_Mjf_Thr).
    *
    The MFPS threshold values.
    *
     ---------- end definition ----------


Minor_Frame_Acct                ( data      ,         )

  = Num_PCD_MNF_Sync_Errors +
    Num_PCD_Filled_MNF +
    Failed_PCD_Votes.
    *
    Statistics gathered from the building of the PCD Minor Frames.
    *

     ---------- end definition ----------


Mission_Start_Time              ( data      ,         )


  = Time.
    *
    The start time of the Landsat Mission.
    *
     ---------- end definition ----------


Mjf_CADU_BCH_Bits_Corr          ( data      ,         )

  = Natural.
    *
    The total number of bits that were BCH corrected in the
    mission data zone of a CADU on a subinterval basis
    *
     ---------- end definition ----------


Mjf_CADU_BCH_Corr_Cnt           ( data      ,         )

  = Natural.
    *
    The number of CADUS with BCH errors corrected in the mission
    data zone per major frame.
    *
     ---------- end definition ----------


Mjf_CADU_BCH_Uncorr_Cnt         ( data      ,         )
```

= Natural.
*
The number of CADUS with BCH errors in the mission data zone uncorrected per major frame.
*
---------- end definition ----------

Mjf_CADU_BER_Cnt (data ,)

= Natural.
*
The BER (based on BCH and/or CRC detected bit errors) in the mission data zone on a subinterval basis.
*
---------- end definition ----------

Mjf_CADU_Bit_Slip (data ,)

= Natural.
*
Indicates the bit slip total for the currently accumulated major frame set.
*
---------- end definition ----------

Mjf_CADU_CRC_Err_Cnt (data ,)

= Natural
*
The count of CADUs with CRC errors.
*.

---------- end definition ----------

Mjf_CADU_Fly_Cnt (data ,)

= Natural.
*
The count of flywheel CADUs for a subinterval.
*
---------- end definition ----------

Mjf_CADU_Missing_Cnt (data ,)

= Natural.
*
Number of missing CADUs per major frame.
*
---------- end definition ----------

Mjf_CADU_Polarity (data ,)

= Natural.
*
The CADU synchronization information polarity for the
currently accumulated set of major frames.
*
---------- end definition ----------

Mjf_CADU_Rcvd_Cnt (data ,)

= Natural.
*
The count of received CADUs per subinterval.
*
---------- end definition ----------

Mjf_CADU_RS_Corr_Cnt (data ,)

= Natural.
*
The number of correctable VCDU headers per major frame.
*
---------- end definition ----------

Mjf_CADU_RS_Uncorr_Cnt (data ,)

= Natural.
*
The number of uncorrectable VCDU headers per major frame.
*
---------- end definition ----------

Mjf_CADU_Seq_Err_Cnt (data ,)

= Natural.
*
The number of VCDU counter errors in a major frame.
*
---------- end definition ----------

Mjf_CADU_Seq_Err_Thr (data ,)

= Natural
*
Threshold value for the number of VCDU sequence counter errors that occur in a subinterval.

*.
---------- end definition ----------

Mjf_CADU_Sync_Err_Cnt (data ,)

= Natural.
*
The count of CADUs with synchronization errors for a major frame.
*
---------- end definition ----------

Mjf_CADU_Sync_Info (data ,)

= Mjf_CADU_Polarity +
Mjf_CADU_Sync_Strategy +
Mjf_CADU_Bit_Slip.
*
The synchronization information including the polarity,
the synchronization strategy and the number of bit slips
for a subinterval.
*
---------- end definition ----------

Mjf_CADU_Sync_Strategy (data ,)

= Valid_CCSDS_Parms.
*
The CADU synchronization strategy setup parameters on a
subinterval basis.
*
---------- end definition ----------

Mjf_Count (data ,)

= Natural.
*
The total number of major frames on a subinterval basis.
*
---------- end definition ----------

Mjf_Data_Rate (data ,)

= Real.
*
The nominal data rate of major frames per millisecond.
*
---------- end definition ----------

```
Mjf_Eol_Err_Cnt                ( data      ,          )

  = Natural.
   *
   The number of end of line errors encountered on a
   subinterval basis.

   *
    ---------- end definition ----------


Mjf_Full_Fill_Cnt              ( data      ,          )

  = Natural.
   *
   The number of major frames that need to be fully filled
   for a subinterval.
   *
    ---------- end definition ----------


Mjf_Part_Fill_Cnt              ( data      ,          )

  = Natural.
   *
   The number of major frames that need to be partially filled
   for a subinterval.
   *
    ---------- end definition ----------


Mjf_QA                         ( data      ,          )

  = Mjf_Count +
    Mjf_Tossed_Cnt +
    Mjf_Eol_Err_Cnt +
    Mnf_Ctr_Err.
   *
   The major frame quality and accounting information
   calculated on a subinterval basis.
   *
    ---------- end definition ----------


MJF_Time_And_Position          ( store     ,          )

  = Sub_Intv_Id +
    {
     @Major_Frame_Time +
     Attitude +
     Ephemeris +
     Cal_Door_Activity_Status
    }.
   *
```

```
    The position and time of each PCD Major Frame within a
    sub-interval.
    *
     ---------- end definition ----------


Mjf_Time_Code_Err_Cnt              ( store      ,          )

  = Natural.
    *
    The count of imagery time code errors on a subinterval basis.
    *
     ---------- end definition ----------


Mjf_Tossed_Cnt                   ( data       ,          )

  = Natural.
    *

    The number of major frames that are calculated from the
    sync errors and the end of line code errors on a subinterval
    basis.
    *
     ---------- end definition ----------


Mjf_VCDU_Cnt_Totals              ( store      ,          )

  = Mjf_QA +
    VCDU_QA +
    Mjf_Full_Fill_Cnt +
    Mjf_Time_Code_Err_Cnt.
    *
    Quality and accounting information totals for the currently processed
    Major Frames and VCDUs in a particular subinterval.
    *
     ---------- end definition ----------


Mjf_VCDU_Data                    ( data       ,          )

  = Gap_Tagged_VCDUs +
    Direction_Start.
    *
    Contains the annotated VCDUs and any VCDUs with gaps tagged.
    The start of either a forward or reverse scan is contained
    in the direction start.
    *
     ---------- end definition ----------


Mjf_VCDU_Parms                   ( data       ,          )
```

```
  = Maj_Vote_Tol +
    Part_Mnf_Tol +
    Mjf_Data_Rate +
    Time_Range_Tol +
    Sub_Intv_Delta.
    *
    MFPS Parameters used during the parsing of the major frames.
    *
     ---------- end definition ----------


Mjf_VCDU_QA                    ( store     ,          )

  = VCDU_QA +
    Mjf_QA +
    Mjf_Time_Code_Err_Cnt +
    Mjf_Full_Fill_Cnt +
    Mjf_Part_Fill_Cnt.
    *
    Currently calculated quality and accounting information
    accumulated for CADUs and major frames on a subinterval
    basis.
    *
     ---------- end definition ----------


Mjf_VCDU_QA_Report_Info            ( data      ,          )

  = Mjf_Count +
    Mjf_CADU_Sync_Info +
    Mjf_CADU_Sync_Err_Cnt +
    Mjf_CADU_Rcvd_Cnt +

    Mjf_CADU_Fly_Cnt +
    Mjf_CADU_Missing_Cnt +
    Mjf_CADU_RS_Corr_Cnt +
    Mjf_CADU_RS_Uncorr_Cnt +
    Mjf_CADU_BCH_Corr_Cnt +
    Mjf_CADU_BCH_Uncorr_Cnt +
    Mjf_CADU_CRC_Err_Cnt +
    Mjf_Full_Fill_Cnt +
    Mjf_Part_Fill_Cnt +
    Mjf_CADU_BER_Cnt.
    *
    The Level 0R quality and accounting data needed to generate
    the Level 0R quality and accounting report on a subinterval
    basis.
    *
     ---------- end definition ----------


Mnf_Ctr_Err                    ( data      ,          )

  = Natural.
```

```
    *
    The number of minor frame counter errors in a major frame
    on a subinterval basis.
    *
     ---------- end definition ----------


Mnf_Ctr_Thr                ( data      ,          )

  = Natural
    *
    Threshold value for the number of minor frame counter errors that in a subinterval.
    *.
     ---------- end definition ----------


Modification_Type          ( data      , discrete    )

  = [
     "INSERT" |
     "UPDATE" |
     "DELETE"
    ].
    *
    Description:
    A database store modification type.
    *

     ---------- end definition ----------


Mono                       ( data      ,          )

  = Integer

    *
    A band parameter that specifies which monochrome band to
    process for Browse and ACCA.
    *.
     ---------- end definition ----------


Mono_Band_Subs             ( data      ,          )

  = Mono_Subs_Image +

    MF_Start_Time +
    MF_Stop_Time +
    Sub_Intv_Id +
    Subs +
    Wave.
    *
    Image data for one predetermined band that has been reduced
    by subsampling. It also contains the sub-interval start and
```

```
    stop times, the subinterval identifier and the subsampling
    and wavelet reduction ratios.
    *
     ---------- end definition ----------


Mono_Band_Wave                ( data      ,          )

  = Mono_Wave_Image +
    MF_Start_Time +
    MF_Stop_Time +
    Sub_Intv_Id +
    Subs +
    Wave.
    *
    Image data from one predetermined band that has been
    reduced first by subsampling and next by wavelets.
    It also contains the subinterval start and stop times,
    the subinterval identifier, and the subsampling and wavelet
    reduction ratios.
    *
     ---------- end definition ----------


Mono_Browse_File_Name         ( data      ,          )

  = File_Location +
    "L7_" +
    MF_Start_Time +
    "_" +
    Mono_Browse_File_Type +
    "_" +
    File_Version_Number.
    *
    Name of monochrome browse file.
    *
     ---------- end definition ----------


Mono_Browse_File_Type         ( data      , discrete )

  = "Mono".
    *
    A string identifying the monochrome browse file.
    *
     ---------- end definition ----------


Mono_Status                   ( data      ,          )

  = Message.
    *
    Status indicating the success or failure of the validation
    of monochrome band parameter.
```

```
    *
     ---------- end definition ----------


Mono_Subs_Image                ( data      ,          )


  = {Pixel}.
    *
    Image data for one predetermined band that has been reduced
    by subsampling.
    *
     ---------- end definition ----------


Mono_Wave_Image                ( data      ,          )

  = {Pixel}.
    *
    Image data from one predetermined band that has been
    reduced first by subsampling and next by wavelets.
    *
     ---------- end definition ----------


MSCD_Data                      ( store     ,          )

  = {
     Scan_Dir +
     SHS_Err +
     FHS_Err
    }.
    *
    The mirror scan correction data collected for the MSCD_File
    on a major frame basis for a given subinterval. The MSCD data is
    located immediately following the End of line code pattern and
    is extracted from 2 contiguous minor frames.
    *
     ---------- end definition ----------


MSCD_File                      ( store     ,          )

  = {
     Scan_Dir +
     SHS_Err +
     FHS_Err
    }.
    *
    A file containing the Scan Line data extracted from the two
    minor frames following the End of Line Code in each major
    frame for a given subinterval.
    *
     ---------- end definition ----------
```

```
MSCD_File_Name                ( data      ,         )

  = File_Location +
    "L7_" +
    MF_Start_Time +
    "_" +
    MSCD_File_Type +
    "_" +
    File_Version_Number.
    *
    The mirror scan correction file name.
    *
     ---------- end definition ----------


MSCD_File_Type                ( data      , discrete    )


  = "MSCD".
    *
    A string identifying the MSCD file.
    *
     ---------- end definition ----------


Multi1                        ( data      ,         )

  = Integer

    *
    A band parameter that specifies the first of three bands
    to process for Browse and ACCA.
    *.
     ---------- end definition ----------


Multi1_Status                 ( data      ,         )

  = Message.
    *
    Status indicating the success or failure of the validation
    of the multiband 1 band parameter.
    *
     ---------- end definition ----------


Multi2                        ( data      ,         )

  = Integer

    *
    A band parameter that specifies the second of three bands
```

to process for Browse and ACCA.
*.
---------- end definition ----------

Multi2_Status (data ,)

= Message.
*
Status indicating the success or failure of the validation
of the multiband 2 band parameter.
*
---------- end definition ----------

Multi3 (data ,)

= Integer

*
A band parameter that specifies the third of three bands
to process for Browse and ACCA.

*.
---------- end definition ----------

Multi3_Status (data ,)

= Message.

*
Status indicating the success or failure of the validation
of the multiband 3 band parameter.
*
---------- end definition ----------

Multi_Band_Subs (data ,)

= Multi_Subs_Image +
MF_Start_Time +
MF_Stop_Time +
Sub_Intv_Id +
Subs +
Wave.
*
Image data from three predetermined bands that has been
reduced by subsampling. It also contains the subinterval
start and stop times, the subinterval identifier, and the
subsampling and wavelet reduction ratios.
*
---------- end definition ----------

```
Multi_Band_Wave                ( data      ,          )

  = Multi_Wave_Image +
    MF_Start_Time +
    MF_Stop_Time +
    Sub_Intv_Id +
    Subs +
    Wave.
    *
    Image data from three predetermined band that has been
    reduced first by subsampling and next by wavelets.
    It also contains the subinterval start and stop times,
    the subinterval identifier, and the subsampling and wavelet
    reduction ratios.
    *
     ---------- end definition ----------


Multi_Browse_File_Name         ( data      ,          )

  = File_Location +
    "L7_" +
    MF_Start_Time +
    "_" +
    Multi_Browse_File_Type +
    "_" +
    File_Version_Number.
    *
    Name of multiband browse file.
    *
     ---------- end definition ----------


Multi_Browse_File_Type         ( data      , discrete    )

  = "Multi".
    *
    A string identifying the multiband browse file.
    *

     ---------- end definition ----------


Multi_Subs_Image               ( data      ,          )


  = {Pixel}.
    *
    Image data from three predetermined bands that has been
    reduced by subsampling.
    *
     ---------- end definition ----------
```

```
Multi_Wave_Image                ( data      ,         )

  = {Pixel}.
   *
   Image data from three predetermined band that has been
   reduced first by subsampling and next by wavelets.
   *
    ---------- end definition ----------


MUX_Id                          ( data      ,         )

  = Bit
   *
   Multiplexer (MUX) assembly ID. Word 7 bits 1-3 of
   PCD/Status Data.
   *
    ---------- end definition ----------


Natural                         ( data      , primitive   )

  = *
   The range 0-MAXINT.
   *
    ---------- end definition ----------


Nominal_Scene_Coords            ( data      ,         )

  = Sub_Intv_Id +
   PCD_Scene_Count +
   Sub_Intv_Scene_Num +
   WRS_Row_Nominal +
   WRS_Path_Nominal +
   Scene_Center_Time +
   Scene_Center_Scan_Num +
   Cal_Door_Activity_Status.
   *
   The parameters describing the nominal position for each
   WRS scene.
   *
    ---------- end definition ----------


Num_Avail_ADP                   ( data      ,         )

  = Natural.
   *
   The number of attitude data points that are available on a sub interval
   basis.
   *
    ---------- end definition ----------
```

```
Num_Avail_EDP                ( data      ,         )

  = Natural.
   *
   The number of ephemeris data points that are available on a sub interval
   basis.
   *
    ---------- end definition ----------


Num_Failed_Votes             ( data      ,         )

  = Integer.
   *
   The threshold for reporting errors when unsuccessful majority votes are
   performed.
   *
    ---------- end definition ----------


Num_Missing_ADP              ( data      ,         )

  = Natural.
   *
   The number of attitude data points that are not
   available for a PCD Major Frame on a sub interval basis.
   *
    ---------- end definition ----------


Num_Missing_Data_Words       ( data      ,         )

  = Integer.
   *
   The threshold for reporting errors when PCD Information Words are missing
   due to missing VCDUs.
   *
    ---------- end definition ----------


Num_Missing_EDP              ( data      ,         )

  = Natural.
   *
   The number of ephemeris data points that are not
   available for a PCD Major Frame on a sub interval basis.
   *
    ---------- end definition ----------


Num_Missing_VCDUs            ( data      ,         )
```

= Natural.
*
The number of missing VCDUs prior to the current VCDU.
*
---------- end definition ----------

Num_PCD_Filled_MJF (data ,)

= Natural.
*
The number of PCD Major Frames that contain fill values for a sub interval.
*
---------- end definition ----------

Num_PCD_Filled_MNF (data ,)

= Natural.
*
The number PCD Minor Frames that contain predefined fill values
on a sub interval basis.
*
---------- end definition ----------

Num_PCD_MJF (data ,)

= Natural.
*
The number of PCD Major Frames per sub interval.
*
---------- end definition ----------

Num_PCD_MNF_Sync_Errors (data ,)

= Natural.
*
The number of missing PCD Minor Frame sync words per sub interval.
*
---------- end definition ----------

Num_Rejected_ADP (data ,)

= Natural.
*
The number of attitude data points that were determined
invalid on a sub interval basis.
*
---------- end definition ----------

```
Num_Rejected_EDP                ( data      ,         )

  = Natural.
   *
   The number of ephemeris data points that were determined
   invalid on a sub interval basis.
   *
    ---------- end definition ----------


Online_Browse_File_Count        ( data      ,         )

  = Natural
   *
   Count of online browse files.
   *
    ---------- end definition ----------


Online_L0R_File_Count           ( data      ,         )

  = Natural
   *
   Count of online L0R files.
   *
    ---------- end definition ----------


Online_LPS_File_Names           ( data      ,         )


  = { Contact_File_Names }.

   *
   The names of LPS output files which are available for transfer
   to the LP DAAC and for which a DAN has been sent.
   *
    ---------- end definition ----------


Online_Metadata_Count           ( data      ,         )

  = Natural
   *
   Count of online metadata files.
   *
    ---------- end definition ----------


Orbit_Num                       ( data      ,         )

  = Natural.
```

```
    *
    The current revolution/orbit count.
    *

     ---------- end definition ----------


Orbit_Time                  ( data      ,          )

  = Time.
    *
    Orbit upper time range.
    *
     ---------- end definition ----------


Overlay_Marks               ( data      ,          )

  = {Scene_Id +
     Pixel_Index
    }.

    *
    Identifies the pixels in Mono_Band_Wave and Multi_Band_Wave
    which contain a scene center id.
    *
     ---------- end definition ----------


Overlay_Subs                ( data      ,          )

  = Overlay_Marks +
    MF_Start_Time +
    MF_Stop_Time +
    Sub_Intv_Id +
    Subs +
    Wave.
    *
    Specifies the overlay tick marks for the data in
    Mono_Band_Subs and Multi_Band_Subs.
    *

     ---------- end definition ----------


Overlay_Wave                ( data      ,          )

  = Overlay_Marks +
    MF_Start_Time +
    MF_Stop_Time +
    Sub_Intv_Id +
    Subs +
    Wave.
    *
```

```
    Specifies the overlay tick marks for the data in
    Mono_Band_Wave and Multi_Band_Wave.
    *
     ---------- end definition ----------


Pan_Det_Data                ( data      ,         )

  = 6313{Byte}6313.
    *
    Aligned detectors for pan data.
    *
     ---------- end definition ----------


Part_Mjf_Thr                ( data      ,         )

  = Natural.
    *
    The threshold value for the number of partially filled major frames
    in a subinterval.
    *
     ---------- end definition ----------


Part_Mnf_Tol                ( data      ,         )

  = Natural
    *
    The minimum number of data word groups required before a partial minor frame is
    considered readable.
    *.
     ---------- end definition ----------


Partial_PCD_Cycle           ( store     ,         )

  = 1{PCD_Major_Frame}3.
    *
    A buffer which contains the PCD Major Frames that will be used
    to build the Full_PCD_Cycle.
    *
     ---------- end definition ----------


Partial_PCD_Major_Frame     ( store     ,         )

  = 1{PCD_Minor_Frame}128.
    *
    A buffer that contains >=1<128 PCD Minor Frames.
    *
     ---------- end definition ----------
```

```
Partial_PCD_Minor_Frame          ( store     ,         )


  = 1{Info_Word}128.
   *
   A buffer that contains >=1<128 PCD information words.
   *

    ---------- end definition ----------


Partial_Rep_Info_Word            ( store     ,         )

  = Info_Word_1
   + Info_Word_2
   + Info_Word_3.
   *
   Buffer for information words that have not been
   declared as Rep_Info_Words.
   *
    ---------- end definition ----------


PCD_Acct                         ( store     ,         )

  = {
    @Sub_Intv_Id +
    PCD_File_Name +
    Num_PCD_MJF +
    First_PCD_MJF_Time +
    Orbit_Num +
    4{
     Minor_Frame_Acct +
     Major_Frame_Acct
    }4 +
    PCD_Scene_Count +
    0{
     Sub_Intv_Scene_Num +
     WRS_Path_Nominal +
     WRS_Row_Nominal +
     Scene_Center_Time +
     Scene_Center_Scan_Num +
      Horizontal_Display_Shift +
     Sun_Azimuth +
     Sun_Elevation +
     Cal_Door_Activity_Status
    }PCD_Scene_Count
   }.
   *
   A file that contains the statistics that are gathered from
   the processing of PCD bytes, the building of PCD Minor and
   Major Frames, and the extraction and interpretation of
   information from the PCD Major Frames. The PCD_Acct file is
   maintained on a subinterval basis.
```

```
    *
     ---------- end definition ----------


PCD_Assemble_Cycle_Status          ( data      ,          )

  = [
     Data_Missing_Message |
     Failed_Votes_Message
    ].
    *
    Errors that are detected during the processing of PCD data.
    *

     ---------- end definition ----------


PCD_Bytes                    ( data      ,          )

  = 4{Byte}4.
    *
    The four bytes that are extracted from the PCD/Status field of the VCDU.
    *
     ---------- end definition ----------


PCD_Directive                ( data      ,          )

  = [
     PCD_Parms |
     PCD_Thresholds
    ].
    *
    The parameters used by the PCDS during the subsystem
    initialization process. The parameters are maintained in
    the PCDS.
    *
     ---------- end definition ----------


PCD_File                     ( store     ,          )

  = {Qualified_PCD_Cycle}.
    *
    A file containing the PCD major frames received during
    a subinterval on a full PCD cycle basis. A PCD cycle
    consists of four PCD major frames.
    *
     ---------- end definition ----------


PCD_File_Info                ( data      ,          )

  = Sub_Intv_Id +
```

```
    PCD_File_Name +
    Num_PCD_MJF +
    First_PCD_MJF_Time +
    Orbit_Num.
    *
    PCD_File related data that identifies the file and describes
    the contents of the file.
    *

     ---------- end definition ----------


PCD_File_Name                ( data      ,          )

  = File_Location +
    "L7_" +
    MF_Start_Time +
    "_" +
    PCD_File_Type +
    "_" +
    File_Version_Number.
    *
    The PCD file name.
    *
     ---------- end definition ----------


PCD_File_Type                ( data      , discrete    )


  = "PCD".
    *
    PCD File identifier string.
    *
     ---------- end definition ----------


PCD_Frame_Fill               ( data      ,          )

  = Byte.
    *
    Predefined value that is stored used to fill missing PCD data
    when building PCD Minor and Major Frames.
    *
     ---------- end definition ----------


PCD_Frame_Info               ( store     ,          )

  = Sub_Intv_Id +
    4{
     Minor_Frame_Acct +
     Major_Frame_Acct
    }4.
```

```
    *
    The statistics that are gathered from the processing of PCD
    bytes and the building of the PCD Major and Minor Frames.
    Information is gathered on a full PCD cycle basis.
    *
     ---------- end definition ----------


PCD_Info                    ( data      ,         )

  = Sub_Intv_Id +
    PCD_Bytes +
    End_Of_Contact_Flag +
    Num_Missing_VCDUs.
    *
    The information needed by the PCDS.
    *
     ---------- end definition ----------


PCD_Info_Word               ( data      ,         )

  = Sub_Intv_Id +
    Info_Word +
    Flush_Minor_Frame +
    Info_Word_Missing +
    Majority_Vote_Failure.
    *
    The information word that will be used to build the PCD Minor Frame(s).
    *
     ---------- end definition ----------


PCD_Major_Frame             ( data      ,         )

  = 128{PCD_Minor_Frame}128.
    *
    A structure that contains 128 PCD Minor Frames.
    *

     ---------- end definition ----------


PCD_Minor_Frame             ( data      ,         )

  = 128{Info_Word}128.
    *
    A structure that contains 128 PCD information words.
    *
     ---------- end definition ----------


PCD_Parms                   ( data      ,         )
```

```
  = (ETM_Plus_To_Body_Trans) +
    (Mission_Start_Time) +
    (Time_Per_Orbit) +
    (ETM_Plus_LOS_x) +
    (ETM_Plus_LOS_y) +
    (ETM_Plus_LOS_z) +
    (Semi_Major_Axis) +
    (Semi_Minor_Axis) +
    (PCD_Frame_Fill) +
    (
     WRS_Row_Nominal +
     Latitude +
     {
      WRS_Path_Nominal +
      Longitude
     }
    ).
    *
    PCD Parameters used in processing PCD data including:
    Parameters that are provided by the IAS and used to
    calculate the longitude and latitude, the WRS Scene Id,
    and sun elevation and azimuth, and
    The Worldwide Reference System parameters containing the
    information for one WRS row.
    *
     ---------- end definition ----------


PCD_Scene_Count                ( store      ,          )

  = Natural.
    *
    The number of WRS scenes that have been identified within a
    sub-interval.
    *
     ---------- end definition ----------


PCD_Setup_Status               ( data       ,          )

  = Message.

    *
    An error message containing a list of the invalid PCD parameters and
    thresholds and their values.
    *
     ---------- end definition ----------


PCD_Status                     ( data       ,          )

  = [

     PCD_Setup_Status |
```

```
    PCD_Assemble_Cycle_Status
   ].
   *
   Status messages sent from the PCDS to the MACS.
   *
    ---------- end definition ----------


PCD_Thresholds                ( data      ,          )

  = (Ephem_Position_Upper) +
   (Ephem_Position_Lower) +
   (Ephem_Velocity_Upper) +
   (Ephem_Velocity_Lower) +
   (Att_Lower_Bounds) +
   (Att_Upper_Bounds) +
   (Num_Missing_Data_Words) +
   (Num_Failed_Votes).
   *
   PCD threshold parameters used in processing PCD data.
   *
    ---------- end definition ----------


Pixel                         ( data      ,          )

  = Byte

   *
   A byte of image data
   *.
    ---------- end definition ----------


Pixel_Index                   ( data      ,          )

  = Integer.

   *
   Specifies the index into Mono_Subs_Image and Multi_Subs_Image
   pixel image.
   *
    ---------- end definition ----------


PN_Decoded_CADU_Bytes             ( data      ,          )

  = 1036{Byte}1036.
   *
   The bytes of a CADU that have been PN decoded.
   *
    ---------- end definition ----------
```

```
Polarity_Unknown_Bytes          ( data      ,       )

  = 1036{Byte}1036
   *
   A stream of bytes from the raw wideband data that has not been
   checked for polarity
   *
    ---------- end definition ----------


Polarity_Unknown_Sync           ( data      ,       )


  = [ Sync |
     Inverted_Sync
   ]
   *
   A sync marker that has not been checked for polarity
   *
    ---------- end definition ----------


Prev_VCID                       ( data      ,       )

  = VCID
   *
   The VCID of the previous VCDU
   *
    ---------- end definition ----------


Previous_Lat_And_Long           ( store     ,       )

  = Lat_And_Long.
   *
   Latitude and longitude values that were previously checked
   against the latitude and longitude values in the WRS scheme.
   The latitude and longitude values are retained to confirm
   WRS scene center crossing and for use in interpolating the
   actual scene center latitude and longitude.
   *
    ---------- end definition ----------


Qualified_PCD_Cycle             ( data      ,       )

  = Full_PCD_Cycle.
   *
   A Full_PCD_Cycle that has been time checked and determined
   to be part of the current sub interval.
   *
    ---------- end definition ----------
```

```
Raw_Data_Byte_Stream              ( data      ,          )

  = {Byte}.
   *
   Byte stream for a single LPS string containing Raw_WB_Data.
   *
    ---------- end definition ----------


Raw_WB_Sets                       ( store     ,          )

  = {
    @Contact_Id +
    Raw_Data_Byte_Stream
   }.
   *
   This store contains the Raw_WB_Data to be processed.
   *
    ---------- end definition ----------


Rcv_Dat_L7_Scenes                 ( data      ,          )

  = Natural.

   *
   Approximate number of scenes in Raw_WB_Data set
   (Rcv_Dat_Vol_Mbytes / Mbytes per scene).
   *
    ---------- end definition ----------


Rcv_Dat_Vol_Mbytes                ( data      ,          )

  = Natural.
   *
   Data volume in megabytes derived from the difference between
   the Contact_Start_Time and the Contact_Stop_Time divided by
   number of megabytes per second.
   *
    ---------- end definition ----------


RDC_Acct                          ( store     ,          )

  = {
    @Contact_Id +
    Rcv_Dat_Vol_Mbytes
   }.
   *
   Raw Data Capture accounting information.
   *
    ---------- end definition ----------
```

```
RDC_Capture_Drct                ( data      , discrete   )

  = [
    "START" |
    "STOP"
   ].
   *
   The directive sent by MACS used to control LPS raw data
   capture process
   *
    ---------- end definition ----------


RDC_Capture_Stat                ( data      ,          )

  = Message.
   *
   A message returned to MACS stating the disposition of
   capture of raw wideband data and a Contact_Id following
   received MACS directive or a warning message stating
   that the datastore already contains three contact periods.
   *
    ---------- end definition ----------


RDC_Delete_Drct                 ( data      ,          )

  = Contact_Id.
   *
   The directive sent by MACS used to delete the Raw_WB_Data
   file for the contact period requested.
   *
    ---------- end definition ----------


RDC_Delete_Stat                 ( data      ,          )


  = Message.
   *
   A message stating the deletion of the Raw_WB_Data file
   is complete.
   *
    ---------- end definition ----------


RDC_Directive                   ( data      ,          )

  = [
    RDC_Capture_Drct |
    RDC_Delete_Drct |
    RDC_Save_Drct |
    RDC_Restage_Drct |
```

```
    RDC_Rpt_Data_Capture_Sum_Drct
   ].
   *
   DESCRIPTION:
   The control directives from LPS personnel to guide the LPS
   data capture processing.
   *
    ---------- end definition ----------


RDC_Restage_Drct              ( data      ,         )

  = Contact_Id +
   [
    "START" |
    "STOP"
   ].
   *
   The directive sent by MACS used to control the restaging of
   the raw wideband data from the 60-day store process.
   *
    ---------- end definition ----------


RDC_Restage_Stat              ( data      ,         )

  = Message.
   *
   A message stating the restaging of a raw
   wideband data set (Raw_WB_Data) is complete.
   *
    ---------- end definition ----------


RDC_Rpt_Data_Capture_Sum_Drct      ( data      ,         )

  = Contact_Id

   *
   DESCRIPTION:
   Upon receipt of this control directive from LPS personnel, the RDCS will
   gather the Landsat 7 raw data capture summary information for
   the operator specified contact period and forward the information to the
   MACS for either display or hardcopy report.  This directive is used by
   an event basis.
   *
    ---------- end definition ----------


RDC_Save_Drct                 ( data      ,         )


  = Contact_Id +
   [
```

```
    "START" |
    "STOP"
   ].
   *
   The directive sent by MACS used to start the LPS raw data
   save to removable media containing the Contact_Id needed to
   identify the data set.
   *
    ---------- end definition ----------


RDC_Save_Stat                ( data      ,          )

  = Message.
   *
   A message sent to the MACS stating the saving of raw
   wideband data is complete.
   *
    ---------- end definition ----------


RDC_Status                   ( data      ,          )

  = [
    RDC_Capture_Stat |
    RDC_Delete_Stat |
    RDC_Save_Stat |
    RDC_Restage_Stat
   ].
   *
   Status messages returned from the RDCS
   *
    ---------- end definition ----------


RDP_Acct                     ( store     ,          )

  = {
    @Contact_Id +
    Valid_CCSDS_Parms +
    CADU_Polarity +
    CADU_Bit_Slip +
    CADU_Sync_Error_Count +
    CADU_Rcv_Count +
    CADU_Flywheel_Count +
    CADU_Missing_Count +
    CADU_CRC_Error_Count +
    VCDU_Header_Correctable_Error_Count +
    VCDU_Header_Uncorrectable_Error_Count +
    BCH_Data_Corrected_Error_Count +
    BCH_Pointer_Corrected_Error_Count +
    BCH_Data_Uncorrected_Error_Count +
    BCH_Pointer_Uncorrected_Error_Count +
    BER
```

```
    }.
    *
    Aggregate information summarizing the CCSDS Grade 3 quality
    of a raw wide band data set.
    *
     ---------- end definition ----------


RDP_BCH_Err_Status              ( data      ,         )


  = Message
    *
    Status message stating BCH errors exceed threshold.
    *

     ---------- end definition ----------


RDP_BER_Err_Status              ( data      ,         )

  = Message.
    *
    Status message stating that the maximum bit error rate has been exceeded.
    *
     ---------- end definition ----------


RDP_CCSDS_Parms                 ( data      ,         )

  = (CADU_Search_Tolerance) +
    (CADU_Check_Tolerance) +
    (CADU_Flywheel_Tolerance) +
    (CADU_Sync_Marker_Check_Error_Tolerance) +
    (CADU_Sync_Lock_Error_Tolerance) +
    (CADU_Bit_Slip_Correction_Extent).
    *
    Parameters that control CCSDS frame synchronization and bit
    slip correction.
    *
     ---------- end definition ----------


RDP_CRC_Err_Status              ( data      ,         )

  = Message
    *
    Status message stating that the CRC error threshold has been exceeded
    *
     ---------- end definition ----------


RDP_Directive                   ( data      ,         )
```

```
  = [
     RDP_Process_Drct |
     RDP_Rpt_Return_Link_QA_Drct |
     RDP_Thresholds |
     RDP_CCSDS_Parms
    ].
    *
    DESCRIPTION:
    The control directives from LPS personnel to guide Landsat
    7 raw data processing.
    *
     ---------- end definition ----------


RDP_Process_Drct              ( data      ,         )

  = Contact_Id.
    *
    Notification to the RDPS to begin processing The contact
    channel information for processing the raw wideband data.
    *
     ---------- end definition ----------


RDP_Rpt_Return_Link_QA_Drct       ( data      ,         )


  = Contact_Id.
    *
    This directive requests a return link summary report for a
    given contact period.
    *

     ---------- end definition ----------


RDP_RS_Err_Status              ( data      ,         )

  = Message
    *
    Status message indicating that the maximum number of Reed_Solomon errors
    has been exceeded.
    *
     ---------- end definition ----------


RDP_Setup_Status              ( data      ,         )

  = Message
    *
    An error message containing a list of the invalid CCSDS parameters and RDP
    thresholds and their values
    *
     ---------- end definition ----------
```

```
RDP_Status                    ( data      ,          )

  = [
     RDP_Setup_Status |
     RDP_Sync_Err_Status |
     RDP_CRC_Err_Status |
     RDP_RS_Err_Status |
     RDP_BCH_Err_Status |
     RDP_BER_Err_Status
    ].
    *
    Status messages returned from the RDPS
    *
     ---------- end definition ----------


RDP_Sync_Err_Status           ( data      ,          )

  = Message
    *
    Status message indicating that the error threshold for sync errors has been
    exceeded.
    *
     ---------- end definition ----------


RDP_Thresholds                ( data      ,          )

  = (Sync_Thres) +
    (CRC_Thres) +
    (RS_Thres) +
    (BCH_Thres) +
    (BER_Thres).

    *
    RDP processing thresholds sent from the MACS.
    *
     ---------- end definition ----------


Real                          ( data      , primitive   )

  = *
    Real number.
    *.
     ---------- end definition ----------


Rel_VCDU_Cnt                  ( data      ,          )

  = Natural.
    *
```

```
    At any time the current number of VCDUs in Ann_VCDU_Collection.
    *
     ---------- end definition ----------


Removable_Media                ( store     ,          )

  = {
    @Contact_Id +
    Raw_Data_Byte_Stream
    }.
    *
    The collection of raw wideband data sets stored on
    removable media.
    *
     ---------- end definition ----------


Rep_Info_Word                  ( data      ,          )

  = Sub_Intv_Id +
    Info_Word_1 +
    Info_Word_2 +
    Info_Word_3 +
    Info_Word_Missing +
    Contact_Ended.
    *
    A structure that contains the PCD_Info_Word in triplicate
    and the PCD Information Word status.
    *
     ---------- end definition ----------


Report_LDT_File_Xfer_Sum       ( data      ,          )

  = Current_Time +
    Date_Of_Report_Data +
    Available_LPS_File_Names +
    Available_L0R_File_Count +
    Available_Browse_File_Count +
    Available_Metadata_Count +
    Online_LPS_File_Names +
    Online_L0R_File_Count +
    Online_Browse_File_Count +
    Online_Metadata_Count +
    Transmitted_LPS_File_Names +
    Volume_Of_Retained_Data +
    Available_Retention_Space.


    *
    A report containing the count of LPS files available,
    the count of LPS files retained online, the count of LPS
    files transmitted to the LP DAAC, and the usage and
```

availability of data output store.
*

---------- end definition ----------

Report_MFP_L0R_QA (data ,)

= Mjf_VCDU_QA_Report_Info +
MF_Start_Time +
MF_Stop_Time.
*
The Level 0R quality and accounting report per subinterval.
*
---------- end definition ----------

Report_RDC_Data_Capture_Sum (data ,)

= Contact_Id +
Rcv_Dat_L7_Scenes +
Rcv_Dat_Vol_Mbytes.
*
Aggregate accounting information for a single channel for a
single contact period.
*
---------- end definition ----------

Report_RDP_Return_Link_QA (data ,)

= LPS_Hardware_String_Id +
Contact_Start_Time +
Contact_Stop_Time +
CADU_Polarity +
CADU_Bit_Slip +
CADU_Sync_Error_Count +
CADU_Rcv_Count +
CADU_Flywheel_Count +
CADU_Missing_Count +
VCDU_Header_Correctable_Error_Count +
VCDU_Header_Uncorrectable_Error_Count +
BCH_Data_Corrected_Error_Count +
BCH_Pointer_Corrected_Error_Count +
BCH_Data_Uncorrected_Error_Count +
BCH_Pointer_Uncorrected_Error_Count +
CADU_CRC_Error_Count +
Approx_Data_Received +
Approx_Major_Frame_Count +
Approx_ETM_Count +
BER.
*
A report containing the RDP_Acct data store's entries for

a specified raw wideband data set (Raw_WB_Data).
*
---------- end definition ----------

RS_Acct (data ,)

= Contact_Id +
VCDU_Header_Correctable_Error_Count +
VCDU_Header_Uncorrectable_Error_Count.
*
The counts of the correctable and uncorrectable header errors determined during the Reed-Solomon error detection and correction.
*
---------- end definition ----------

RS_Annotation (data ,)

= Boolean.
*
An indication of the quality of the associated CADU based on the Reed-Solomon EDAC checks.
*
---------- end definition ----------

RS_Corr_CADU (data ,)

= Contact_Id +
Sync +
[
VCDU_Hdr_Bytes |
VCDU_Fill_Hdr_Bytes
] +
VCDU_Data +
VCDU_Trailer +
Sync_Annotation +
CRC_Annotation +
RS_Annotation
*
A CADU that has completed the Reed_Solomon EDAC and the data quality indicators from the frame sync CRC and Reed_Solomon processes.

NOTE:
The contact ID is not part of the annotation, but simply shown as information necessary to associate this CADU with a Contact_Id.
*
---------- end definition ----------

```
RS_Failed_CADU                ( data      ,          )

  = CRC_Chkd_CADU.

    *
    This is a CADU which has failed the Reed-Solomon checks.
    *
     ---------- end definition ----------


RS_Thres                      ( data      ,          )

  = Natural
    *
    The maximum number of Reed_Solomon errors allowed before notifying the
    operator.
    *
     ---------- end definition ----------


Saved_Time                    ( store     ,          )


  = Actual_Time.
    *
    The storage for the most recent actual major frame time.
    *
     ---------- end definition ----------


Saved_VCID                    ( store     ,          )

  = [
     Prev_VCID |
     Curr_VCID
    ].
    *
    The VCID which is checked for a changed value
    *
     ---------- end definition ----------


Scan_Dir                      ( data      ,          )

  = {
     Char
    }.
    *
    1 byte character.
    *
     ---------- end definition ----------
```

```
Scan_Dir_Thr                    ( data      ,          )

  = Natural.
   *
   Threshold value for the number of Failed_Mjf_VCDU_Set.
   *

    ---------- end definition ----------


Scene_Center_Scan_Num           ( data      ,          )

  = Natural.
   *
   The scan count, within a sub-interval, of the WRS scene
   center.
   *
    ---------- end definition ----------


Scene_Center_Time               ( data      ,          )

  = Time.
   *
   Description:
   The WRS scene center time when the image was taken. The
   information is determined by PCDS and is given to MACS for
   inclusion of metadata file via PCD_Acct.
   *
    ---------- end definition ----------


Scene_Data                      ( data      ,          )


  = {Pixel}.
   *
   Data for a full scene, all bands for format 1
   *
    ---------- end definition ----------


Scene_Data_Store                ( store     ,          )

  = {
    Aligned_Bands +
    Scene_Data
   }.
   *
   Storage where aligned bands accumulates until a
   full scene has been collected.
   *
    ---------- end definition ----------
```

```
Scene_Id                    ( store     ,          )

  = WRS_Row_Nominal +
    WRS_Path_Nominal +
    Sun_Elevation +
    Scene_Center_Time.
    *
    The position, time and sun elevation for each WRS scene.
    *
     ---------- end definition ----------


Scene_Info                  ( data      ,          )

  = PCD_Scene_Count +
    PCD_Scene_Count{
     Scene_Id
    }PCD_Scene_Count.
    *
    A structure that contains the scene identification and the
    parameters used to calculate the scene id for each identified
    scene. The scene identification is in accordance with the
    Worldwide Reference System (WRS) scheme.
    *
     ---------- end definition ----------


Scene_Metadata              ( data      ,          )

  = Sub_Intv_Id +
    Scene_Id
    *
    Uniquely identifies a scene by using the sub-interval from
    which the data came and the scene id itself.
    *
     ---------- end definition ----------


Scene_Parms                 ( store     ,          )

  = Sub_Intv_Id +
    PCD_Scene_Count +
    0{
     Sub_Intv_Scene_Num +
     WRS_Path_Nominal +

     WRS_Row_Nominal +
     Scene_Center_Time +
     Scene_Center_Scan_Num +
     Horizontal_Display_Shift +
     Sun_Azimuth +
     Sun_Elevation +
     Cal_Door_Activity_Status
```

```
    }PCD_Scene_Count.
    *
    Input parameters used to determine the scene id, and the
    intermediate values derived from the scene id calculation.
    There is one entry for each Scene.
    *
     ---------- end definition ----------


SCID                        ( data      ,         )

  = 8{Bit}8.
    *
    Spacecraft ID as it comes in the raw wideband data.
    *
     ---------- end definition ----------


Semi_Major_Axis             ( data      ,         )

  = Real.
    *
    The distance from Apogee or Perigee to the center of the
    orbit ellipse = R(Apogee) + R(Perigee)/2.
    *
     ---------- end definition ----------


Semi_Minor_Axis             ( data      ,         )

  = Real.
    *
    Polar axis radius.
    *
     ---------- end definition ----------


Sensor_Alignment_Info       ( data      ,         )

  = 4{Natural}4.
    *
    Information provided by IAS and used by the MFPS to perform
    integer-pixel alignment.
    *
     ---------- end definition ----------


SHS_Err                     ( data      ,         )

  = 12{
     Char
    }12.
    *
    The second half scan time error.
```

```
    *
     ---------- end definition ----------


Shtr                        ( data      ,          )


  = Bit.
    *
    Shtr 1/2. Word 7 bit 6 of PCD/Status Data.
    Cal Shtr = "0", Backup Shtr = "1".
    *
     ---------- end definition ----------


Spacecraft_Id               ( data      ,          )

  = Natural.
    *
    Landsat 7 spacecraft ID.
    *
     ---------- end definition ----------


Starting_Row                ( data      ,          )

  = WRS_Row_Nominal.
    *
    The beginning row of the first scene in a sub-interval. The
    information is determined by PCDS and is given to MACS for
    inclusion of metadata file via PCD_Acct.
    *
     ---------- end definition ----------


Status_Info                 ( data      ,          )

  = MUX_Id +
    Shtr +
    Format_Id +
    Band_Num +
    Band_Gains +
    Gain_Change_Flag.
    *
    The status information extracted from the PCD/Status field
    of the VCDU. Word 7 and 8 of the PCD/Status Data.
    *
     ---------- end definition ----------


String                      ( data      ,          )

  = {Char}.
    *
```

```
    An array of characters
    *
     ---------- end definition ----------


Sub_Intv                    ( store     ,         )

  = {
    Contact_Id +
    {
     @Sub_Intv_Id +
     MF_Start_Time +
     MF_Stop_Time +
     VCID
    }
   }.
   *
   The beginning and ending major frame times corresponding to

   a predefined subinterval time range for each contact.
   *
    ---------- end definition ----------


Sub_Intv_Delta              ( store     ,         )

  = Real.
   *
   Time in milliseconds.  The Sub_Intv_Delta value after
   validation.
   *
    ---------- end definition ----------


Sub_Intv_File_Names         ( data      ,         )

  = Metadata_File_Name +
   PCD_File_Name +
   Browse_File_Names +
   Cal_File_Name +
   MSCD_File_Name +
   Band_File_Names

   *
   A list of the specific file names associated with a subinterval.
   *.
    ---------- end definition ----------


Sub_Intv_Id                 ( data      ,         )

  = Natural.
   *
   The subinterval's id used to determine the subinterval.
```

*
---------- end definition ----------

Sub_Intv_Info (data ,)

= Contact_Id +
VCID +
VCID_Change_Flag +
End_Of_Contact_Flag.
*
The flags are needed for subinterval determination.
The IDs are needed for the subinterval structure.
*
---------- end definition ----------

Sub_Intv_Scene_Num (data ,)

= Natural.
*
A scene identifier representing the number of scenes identified
thus far in a Sub_Intv.
*
---------- end definition ----------

Subcomm_Word (data ,)

= Byte.
*

Word 72 of all PCD minor frames. Contains data essential to
the ground segment image processing such as Ground
Reference, spacecraft id and various temperatures.
*
---------- end definition ----------

Subs (data ,)

= Natural.
*
A reduction ratio to be used when subsampling browse image
data. Specifies the size square grid to reduce to one
pixel.
*
---------- end definition ----------

Subs_Status (data ,)

= Message.
*

```
    Status indicating the success or failure of the validation
    of subsampling ratio band parameter.
    *
     ---------- end definition ----------


Sun_Azimuth                 ( data      ,         )

  = Real.
    *
    Solar angle.
    *
     ---------- end definition ----------


Sun_Elevation               ( data      ,         )

  = Real.
    *
    Solar elevation.
    *
     ---------- end definition ----------


Sync                        ( data      ,         )

  = 4{Byte}4
    *
    Hex '1ACFFC1D'
    *.
     ---------- end definition ----------


Sync_Annotation             ( data      ,         )

  = CADU_Polarity_Flag +
    CADU_Bit_Slip +
    CADU_Sync_Error_Flag +
    CADU_Flywheel_Flag +
    End_Of_Contact_Flag.
    *
    The data quality indicators from the frame synchronization
    process and a flag to indicate if the end of the contact
    period has been reached.

    *
     ---------- end definition ----------


Sync_Loc                    ( data      ,         )

  = Natural
    *
    The location of the frame sync marker relative to the start of the data
```

*
---------- end definition ----------

Sync_Thr (data ,)

= Natural
*
The threshold value for the number of major frame sync errors occurring in a subinterval.
*.
---------- end definition ----------

Sync_Thres (data ,)

= Natural.
*
The maximum number of sync errors allowed before notifying the operator
*
---------- end definition ----------

Sync_WB_Data (data ,)

= Contact_Id +
Sync_Loc +
0{Bit}8 +
Polarity_Unknown_Sync +
Polarity_Unknown_Bytes +
0{Bit}7 +
Sync_Annotation.
*
The raw wideband data bytes that have not been checked for
polarity, along with the location of the sync marker
and the sync annotations.

NOTE:
The contact ID is not part of the annotation, but simply
shown as information necessary to associate this CADU with
a Contact_Id.
*
---------- end definition ----------

Sync_Word (data ,)

= Byte.
*
Sync pattern, xFAF320, which will appear in words 0 through
2 of each PCD minor frame.
*
---------- end definition ----------

```
Tc_Thr                    ( data      ,          )

  = Natural

    *
    The threshold value for the number of time code errors occurring in a subinterval .
    *.
     ---------- end definition ----------


Time                      ( data      , discrete    )

  = [
     "YYYY DDD HH:MM:SS.mmm.nnn.uuu" |
     "DDD HH:MM:SS.mmm" |
     "time_t" |
     "TBD"
    ].
    *
    LPS time formats.
    *
     ---------- end definition ----------


Time_Available            ( data      ,          )

  = Date +
    Time.

    *
    This object represents the time that the output files associated
    with a particular contact ID were made available for transfer.
    *
     ---------- end definition ----------


Time_Code                 ( data      ,          )

  = {Byte}.
    *
    The VCDUs containing the time code minor frames.
    *
     ---------- end definition ----------


Time_Deleted              ( data      ,          )

  = Date +
    Time.

    *
    This object represents the time that the output files associated
    with a particular contact ID were deleted.
    *
```

---------- end definition ----------

Time_Per_Orbit (data ,)

= Time.
*
The amount of time required for Landsat to make one complete orbit.
*
---------- end definition ----------

Time_Range_Tol (data ,)

= Real.

*
Time in 1/16th of a millisecond. The time range from system time backwards to system time minus the Time_Range_Tol.
*
---------- end definition ----------

Transfer_Request (data , primitive)

=
*
This object represents a list of LPS output files the LP DAAC is requesting from LPS.
*.
---------- end definition ----------

Transmitted_LPS_File_Names (data ,)

= { Contact_File_Names }.

*
The names of LPS output files which have been transmitted to the LP DAAC.
*
---------- end definition ----------

Upper_Left_Corner_Latitude (data ,)

= Real

*
WRS scene upper left corner latitude. The angular distance, measured in degrees, north or south from the equator.
*
---------- end definition ----------

```
Upper_Left_Corner_Longitude          ( data      ,          )

  = Real

    *
    WRS scene upper left corner longitude. Distance east or west on the earth's
    surface, measured as an arc of the equator between the meridian passing
    through a particular place and standard meridian.
    *
     ---------- end definition ----------


Upper_Right_Corner_Latitude          ( data      ,          )

  = Real

    *
    WRS scene upper right corner latitude. The angular distance, measured in
    degrees, north or south from the equator.
    *
     ---------- end definition ----------


Upper_Right_Corner_Longitude         ( data      ,          )

  = Real


    *
    WRS scene upper right corner longitude. Distance east or west on the earth's
    surface, measured as an arc of the equator between the meridian passing
    through a particular place and standard meridian.
    *.
     ---------- end definition ----------


Valid_Band_Parms                     ( store     ,          )

  = Mono +
    Multi1 +
    Multi2 +
    Multi3 +
    Subs +
    Wave +
    CCA_Method +
    CCA_Ratio.
    *
    A datastore containing four validated band parameters
    entered by an operator that specify which bands to
    process for ACCA and Browse.  Valid band numbers are
    1-6. The Mono parameter is for monochrome browse data;
    Multi1, Multi2, and Multi3 are for multiband browse
```

```
    data. Subs and Wave are subsampling and wavelet
    reduction ratios, respectively. CCA_Method and CCA_Ratio
    are two user-defined ACCA parameters.
    *
     ---------- end definition ----------


Valid_CCSDS_Parms                ( store     ,          )

  = CADU_Search_Tolerance +
    CADU_Check_Tolerance +
    CADU_Flywheel_Tolerance +
    CADU_Sync_Marker_Check_Error_Tolerance +
    CADU_Sync_Lock_Error_Tolerance +
    CADU_Bit_Slip_Correction_Extent.
    *
    Validated CCSDS parameters.
    *
     ---------- end definition ----------


Valid_MFP_Parms                  ( store     ,          )

  = Sensor_Alignment_Info +
    Fill_Value +
    Sub_Intv_Delta +
    Mjf_Data_Rate +
    Max_Alignment_Value +
    Time_Range_Tol +
    Part_Mnf_Tol +
    Maj_Vote_Tol.
    *
    The validated MFPS setup parameters.
    *
     ---------- end definition ----------


Valid_MFP_Thres                  ( store     ,          )

  = Mjf_CADU_Seq_Err_Thr +
    Scan_Dir_Thr +
    Sync_Thr +

    Mnf_Ctr_Thr +
    Eol_Thr +
    Tc_Thr +
    Full_Mjf_Thr +
    Part_Mjf_Thr.
    *
    The validated MFPS threshold values.
    *
     ---------- end definition ----------
```

```
Valid_PCD_Parms                ( store     ,         )

  = PCD_Frame_Fill.
   *
   The validated PCD Parameters used in processing PCD data.
   *
    ---------- end definition ----------


Valid_PCD_Thres                ( store     ,         )

  = Ephem_Position_Upper +
   Ephem_Position_Lower +
   Ephem_Velocity_Upper +
   Ephem_Velocity_Lower +
   Att_Lower_Bounds +
   Att_Upper_Bounds +
   Num_Missing_Data_Words +
   Num_Failed_Votes.
   *
   The validated PCD threshold parameters used in processing
   PCD data.
   *
    ---------- end definition ----------


Valid_RDP_Thres                ( store     ,         )

  = Sync_Thres +
   CRC_Thres +
   RS_Thres +
   BCH_Thres +
   BER_Thres
   *
   Validated RDP processing thresholds
   *
    ---------- end definition ----------


Valid_Scene_Parms              ( store     ,         )

  = ETM_Plus_To_Body_Trans +
   Mission_Start_Time +
   Time_Per_Orbit +
   Semi_Major_Axis +
   Semi_Minor_Axis +
   ETM_Plus_LOS_x +
   ETM_Plus_LOS_y +
   ETM_Plus_LOS_z.
   *
   The validated general mission information and parameters
   that are provided by the IAS and used to calculate the
   longitude and latitude, the WRS Scene Id, and sun elevation
   and azimuth.
```

```
    *
     ---------- end definition ----------


Valid_WRS_Parms                ( store     ,         )

  = {
    @WRS_Row_Nominal +
    @WRS_Path_Nominal +
    Center_Latitude +
    Center_Longitude +
    Upper_Left_Corner_Latitude +
    Upper_Left_Corner_Longitude +
    Upper_Right_Corner_Latitude +
    Upper_Right_Corner_Longitude +
    Lower_Left_Corner_Latitude +
    Lower_Left_Corner_Longitude +
    Lower_Right_Corner_Latitude +
    Lower_Right_Corner_Longitude
   }.
    *
    The validated Worldwide Reference System table containing
    the information for each WRS scene.
    *
     ---------- end definition ----------


VCDU_Bytes                     ( data      ,         )

  = 1034{Byte}1034.
    *
    The bytes of a VCDU minus the trailer.
    *
     ---------- end definition ----------


VCDU_Corrected_Mission_Data          ( data      ,         )

  = 7936{Bit}7936.
    *
    The mission data field of the VCDU with 1 to 3 bits BCH corrected
    *
     ---------- end definition ----------


VCDU_Data                      ( data      ,         )

  = VCDU_Mission_Data +
    272{Bit}272.
    *
    A VCDU with an uncorrected mission data zone.
    *
     ---------- end definition ----------
```

```
VCDU_Fill_Hdr_Bytes              ( data      ,         )

  = Bit +
    Fill_SCID +
    54{Bit}54.
    *
    The header portion of a VCDU with a SCID that indicates fill data.
    *
     ---------- end definition ----------


VCDU_Hdr_Bytes                   ( data      ,         )


  = Bit +
    SCID +
    54{Bit}54.
    *
    The header portion of a VCDU.
    *
     ---------- end definition ----------


VCDU_Hdr_Err_Count               ( data      ,         )

  = Mjf_CADU_RS_Corr_Cnt +
    Mjf_CADU_RS_Uncorr_Cnt.
    *
    The total number of VCDU header error control errors
    (Reed-Solomon).
    *
     ---------- end definition ----------


VCDU_Hdr_Fmt1_Correctable_Err_Cnt    ( data      ,         )

  = Natural.
    *
    Count of correctable VCDU headers for format 1.
    *
     ---------- end definition ----------


VCDU_Hdr_Fmt2_Correctable_Err_Cnt    ( data      ,         )

  = Natural.
    *
    Count of correctable VCDU headers for format 2.
    *
     ---------- end definition ----------
```

VCDU_Header_Correctable_Error_Count (data ,)

= VCDU_Hdr_Fmt1_Correctable_Err_Cnt +
VCDU_Hdr_Fmt2_Correctable_Err_Cnt.
*
Count of correctable VCDU headers by VCID.
*
---------- end definition ----------

VCDU_Header_Uncorrectable_Error_Count (data ,)

= Natural.
*
Number of uncorrectable errored VCDUs.
*
---------- end definition ----------

VCDU_Mission_Data (data ,)

= 7936{Bit}7936.
*
The mission data field of the VCDU with no corrections applied.
*
---------- end definition ----------

VCDU_QA (data ,)

= Mjf_CADU_Rcvd_Cnt +
Mjf_CADU_Fly_Cnt +
Mjf_CADU_Sync_Info +
Mjf_CADU_Sync_Err_Cnt +
Mjf_CADU_Missing_Cnt +
Mjf_CADU_RS_Corr_Cnt +
Mjf_CADU_RS_Uncorr_Cnt +
Mjf_CADU_BCH_Corr_Cnt +
Mjf_CADU_BCH_Uncorr_Cnt +
Mjf_CADU_BCH_Bits_Corr +
Mjf_CADU_CRC_Err_Cnt +
Mjf_CADU_BER_Cnt +
Mjf_CADU_Seq_Err_Cnt +
ETM_Data_Format.
*
Currently calculated quality and accounting information
accumulated for CADUs on a subinterval basis.
*
---------- end definition ----------

VCDU_Trailer (data ,)

= 2{Byte}2.
*
The bytes of a VCDU trailer.
*
---------- end definition ----------

VCDU_With_Fill_Info (data ,)

= Ann_VCDU +
Num_Missing_VCDUs.
*
A VCDU structure containing the pointer to the VCDU and the
number of missing VCDUs that were expected before this VCDU.
*
---------- end definition ----------

VCID (data ,)

= 6{Bit}6.
*
The virtual channel Id number.
*
---------- end definition ----------

VCID_Change_Flag (data ,)

= Boolean.
*
Flag indicating whether the VCID has changed
*
---------- end definition ----------

Volume_Of_Retained_Data (data ,)

= Natural.
*

This object represents the number of units of volume of disk
space which are currently being used for LPS output files
retained on-line.
*
---------- end definition ----------

Wave (data ,)

= Natural.
*
A ratio to be used when reducing browse image data by
wavelets. Specifies the size square grid to reduce to one

```
    pixel.
   *
    ---------- end definition ----------


Wave_Status                ( data      ,          )

  = Message.
   *
   Status indicating the success or failure of the validation
   of wavelet ratio band parameter.
   *
    ---------- end definition ----------


WRS_Path_Nominal           ( data      ,          )

  = Natural.
   *
   Standard designator for a nominal scene center.  The
   WRS Path is the east to west index of the WRS Table.
   *
    ---------- end definition ----------


WRS_Row_Nominal            ( data      ,          )

  = Natural.
   *
   Standard designator for a nominal scene center.
   The WRS Row is the north to south index of the WRS Table.
   *
    ---------- end definition ----------
```

Appendix C - WRS Scene Identification Algorithms

This appendix describes and analyzes a WRS scene identification algorithm intended for implementation in the Landsat Processing System (LPS). The appendix provides the following information.

- A summary of the algorithm's input data and their sources.
- A description of the algorithm.
- An analysis of the algorithm's computational complexity.
- An estimate of the Delivered Source Instructions (DSIs) required to implement the algorithm.

Table C-1 PCD Items Needed for WRS Scene Identification

Data	Location	
	Major Frame	Minor Frame
Euler Parameters	0	0 - 15
	1	0 - 15
	2	0 - 15
	3	0 - 15
Spacecraft Position	0	50 - 61
	1	16 - 27
	2	50 - 61
	3	16 - 27
Spacecraft Time	0	28 - 34
	0	96 - 102

Table C-2 IAS Data Items Needed for WRS Scene Identification

Earth semimajor axis
Earth semiminor axis
Transformation matrix from ETM+ line of sight at center of scan to spacecraft body
ETM+ line of sight vector at center of mirror scan

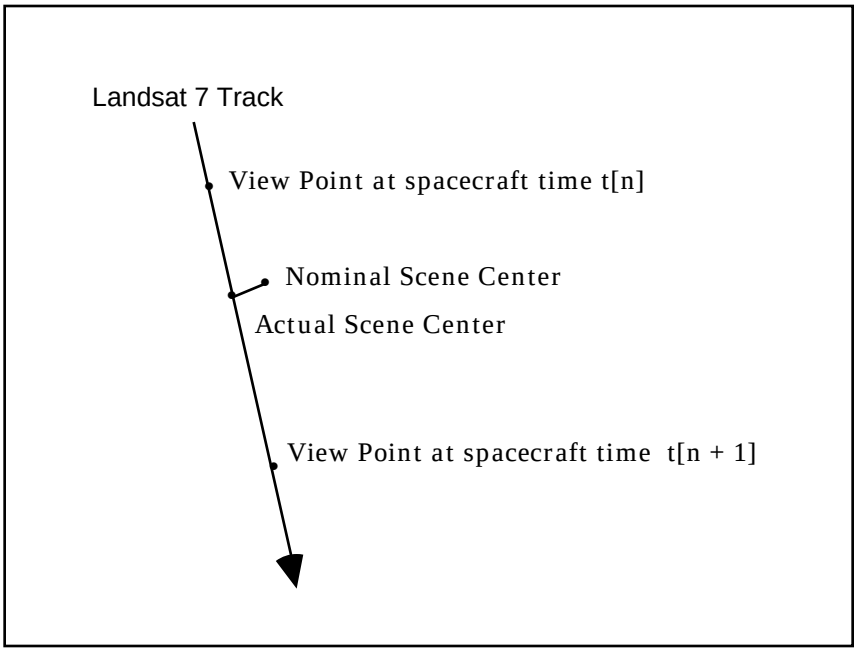
The problem to be addressed is this. Given a continuous sequence of (validated and possibly corrected) 3-tuples containing spacecraft time, attitude, and ephemeris, provide the following information for each WRS scene contained within the sub-interval.

- The scene's WRS path and row.
- The scene center time, i.e. the time code of the major frame containing the datum nearest the nominal WRS scene center position.
- The scene center scan number, i.e. the index (beginning at 1) within the sub-interval's sequence of major frames of the major frame containing the datum nearest the nominal scene center.
- The nominal latitudes and longitudes for the WRS scene center and four corners.
- The Sun azimuth and elevation at the ground point viewed by the datum nearest the nominal WRS scene center position.
- The horizontal display shift, i.e. the distance (in TBD units of measure) between the ground point viewed by the datum nearest the nominal WRS scene center position and the nominal WRS scene center position.

Briefly stated, the algorithm below computes a position (latitude/longitude) of the ground point being viewed for each attitude/ephemeris pair extracted from the PCD. Each position and

its predecessor is used to determine whether the spacecraft passed over a WRS scene center between the two positions (see figure 1). The algorithm uses a look-up table (WRS_LUT) that contains, among other things, the nominal scene center position for each WRS scene. If so, the algorithm interpolates to compute the "actual" scene center's time and position. Because Landsat 7's track is not identical to a track along the WRS scene path, there can be some distance between the nominal scene center and any ground point observed on the track. The actual scene center is then that ground point closest to the nominal scene center defined by the WRS.

Figure C-1 Spacecraft view points bracketing a nominal scene center and the computed actual scene center



The detailed description of the algorithm presented below is divided into three parts. Algorithm #1, "WRS Scene Identification," describes the top-level algorithm for identifying a WRS scene. Separate descriptions are provided for the following components of the algorithm.

- Computing the latitude/longitude of a ground point. This is described in the Latitude and Longitude Computation.

- Computing the Sun azimuth and elevation at a ground point at a given time. This is described in the white paper, *Sun Azimuth and Elevation Algorithms,* January 23, 1995.

- Computing the Geocentric Inertial (GCI) sun vector for a given position at a given time. This computation is required by the Sun azimth and elevation algorithm. It is described in the white paper, *Sun Azimuth and Elevation Algorithms,* January 23, 1995.

- Computing the Greenwich Hour Angle (GHA) for a given position at a given time. This is described in the white paper, *Sun Azimuth and Elevation Algorithms,* January 23, 1995.

C.1 Algorithms

This section describes algorithms used for LPS processing.

Algorithm #1: WRS Scene Identification

This is the top-level algorithm for identifying the center of a WRS scene (and by inference the remainder of the scene) within a Landsat 7 sub-interval. The problem it addresses is as follows. Given a sequence of {spacecraft time (t), attitude (A), ephemeris (X)} 3-tuples extracted from the sub-interval's PCD major frames (with all values corrected, time values computed from the PCD major frame time code, and all values converted to double precision real), and a WRS scene information look-up table (WRS_LUT - described in detail below), identify the scene center of each full WRS scene contained within the sub-interval and output a record containing the scene information required for the sub-interval's metadata file.

Let WRS scene look-up table (WRS_LUT) be a table with entries of the form:

(center_lat/lon, corner_lat/lons, path, row)

where

center_lat/lon = nominal latitude and longitude of WRS scene center

corner_lat/lons = nominal latitude and longitude at each WRS scene corner

path = the path number of this scene

row = the row number of this scene.

Discard the first N - 1{time, attitude, ephemeris} 3-tuples, where N = the minimum number of 3-tuples before encountering the scene center of a full scene within a sub-interval;

t_{prev} <- the spacecraft time of the first 3-tuple in the reduced sequence;

P_{prev} <- the latitiude/longitude of the first 3-tuple in the reduced sequence computed by Algorithm #2;

FOR each remaining {time, attitude, ephemeris} 3-tuple DO

t_{cur} <- the spacecraft time of the 3-tuple;

P_{cur} <- the latitude/longitude of this 3-tuple computed by Algorithm #2;

IF $P_{prev} < C \leq P_{cur}$ for a nominal scene center lat/lon (C) in WRS_LUT THEN

t_{center} <- time of datum nearest the nominal scene center by a TBD interpolation

method;

P_{center} <- latitude/longitude of datum nearest the nominal scene center by a TBD interpolation method;

Horizontal display shift (*HDS*) <- distance between P_{center} and C by a TBD algorithm;

Major_Frame_Time <- the time code of the major frame containing the datum nearest the nominal scene center by a TBD method.

Center_Scan <- the index (beginning at 1) of the major frame containing the datum nearest the nominal scene center.

Compute the Sun azimuth (*AZ*) and elevation (*EL*) at position P_{center} and time t_{center} using the algorithm described in *Sun Azimuth and Elevation Algorithms*, January 23, 1995;

Insert Major_Frame_Time, Center_Scan, *HDS*, *C*, *AZ*, *EL*, WRS_LUT[*C*].corner_lat/lons, WRS_LUT[*C*].path, and WRS_LUT[*C*].row to the metadata for this scene.

END IF;

t_{prev} <- t_{cur} ;

P_{prev} <- P_{cur} ;

END FOR

Algorithm #2: Longitude and Latitude Algorithm

Latitude and Longitude Computation

The problem addressed by this algorithm is the following. Given spacecraft time (t), ephemeris ($\vec{X}$ - Geocentric Inertial (GCI) spacecraft position vector), quaternion [($q1$, $q2$,,$q3$, $q4$) - Euler parameters], Earth semimajor axis (a) and semiminor axis (b), transformation from ETM+ to spacecraft body matrix ($[T_{BM}]$), and ETM+ line of sight vector at the center of the mirror scan (V) , compute the geodetic latitude and longitude of the ground point in the ETM+'s view.

1. Compute the attitude matrix (transformation from GCI to body) from the Euler parameters (quaternion) obtained from PCD.

$$[Q] = \begin{vmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 + q_3q_4) & 2q_1q_3 - q_2q_4 \\ 2(q_1q_2 - q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 + q_1q_4) \\ 2(q_1q_3 + q_2q_4) & 2(q_2q_3 - q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{vmatrix}$$

2. Compute the transformation from spacecraft body to GCI (inverse of $[Q]$)

$$[T_{IB}] = [Q]^{-1}$$

3. Compute the transformation from ETM+ to GCI

$$[T_{IM}] = [T_{IB}][T_{BM}]$$

4. Compute the cross-scan angle (σ) and along-scan angle (θ) of the ETM+ line of sight vector at the center of the mirror scan (V).

5. Solve the following vector equation to obtain the geocentric latitude (φ), right ascension (λ), and slant range (d) to the view point on the ground (see description of this in info provided by T. Keller for subroutine LPC).

$$r\begin{bmatrix} \cos(\varphi)\cos(\lambda) \\ \cos(\varphi)\sin(\lambda) \\ \sin(\varphi) \end{bmatrix} = \vec{X} + d[T_{IM}]\begin{bmatrix} -\sin(\sigma) \\ -\cos(\sigma)\sin(\theta) \\ \cos(\sigma)\cos(\theta) \end{bmatrix}$$

where r is the radius from the center of the Earth to the view point.

6. Compute the Greenwich Hour Angle (GHA, right ascension of Greenwich) at the specified time using subroutine JGHAX.

7. Compute the longitude of the view point from right ascension and GHA.

$$lon = \lambda - GHA$$

8. Compute the geodetic latitude from the geocentric latitude.

$$lat = \arctan[(a/b)^2 \tan(\varphi)]$$

where a and b are the Earth semimajor and semiminor axes (equatorial and polar radii), respectively.

Algorithm #3: Sun azimuth and elevation algorithm

Algorithm Description

The problem to be addressed is this. Given the latitude (*lat*), longitude (*lon*), and spacecraft time (*t*)of the WRS scene center, compute the Sun azimuth (AZ) and elevation (EL) at the ground point. The algorithm is as follows.

1. Compute the Greenwich Hour Angle (GHA, right ascension of Greenwich) at time *t* using subroutine JGHAX.

2. Compute the GCI Sun vector ($\vec{R}_{Sun}$) at time *t* using subroutine SOL.

3. Define an Earth-fixed coordinate system centered at the latitude (*lat*)and longitude (*lon*) of interest with coordinate axes pointing north (N), east (E), and local vertical (V)

$$\hat{V} = \begin{vmatrix} \cos(lon)\cos(lat) \\ \sin(lon)\cos(lat) \\ \sin(lat) \end{vmatrix} \qquad \hat{E} = \begin{vmatrix} -\sin(lon) \\ \cos(lon) \\ 0 \end{vmatrix} \qquad \hat{N} = \begin{vmatrix} -\cos(lon)\sin(lat) \\ -\sin(lon)\sin(lat) \\ \cos(lat) \end{vmatrix}$$

4. Compute the transformation matrix from Earth-fixed to GCI

$$[G] = \begin{vmatrix} \cos(GHA) & -\sin(GHA) & 0 \\ \sin(GHA) & \cos(GHA) & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

5. Transform the Earth-fixed coordinate axes to inertial

$$\hat{V}_N = [G]\hat{N} \qquad \hat{V}_E = \dot{G}]\hat{E} \qquad \hat{V}_V = [G]\hat{V}$$

6. Compute the azimuth and elevation (AZ, EL) of the Sun.

$$AZ = \arctan\left(\frac{\hat{V}_N \cdot \hat{R}_{Sun}}{\hat{V}_E \cdot \hat{R}_{Sun}}\right) \qquad EL = \arcsin\left(\hat{V}_V \cdot \hat{R}_{Sun}\right)$$

C.2 Computational Complexity

The complexity of algorithm depends primarily on the efficiency of the search for nominal scene center positions in WRS_LUT. Although the algorithm describes a complete search at every iteration, the fact that, once a single scene center is identified, each successive scene center is also identified, allows the algorithm to reduce the search for a successive nominal scene centers after the first to a constant.

With this improvement the complexity is proportional to the number of {time, attitude, ephemeris} 3-tuples processed (i.e. O(N)). For each 3-tuple, the cost of the double precision floating point computations required will dominate. The algorithm requires a latitude and longitude computation for each candidate 3-tuple within the sub-interval and two double precision tests (less-than, less-than-or-equal-to). The latitude/longitude computation requires the computation of the GHA. For each identified scene center, the algorithm requires a Sun azimuth and elevation computation. This computation itself requires the computation of a GCI sun vector and GHA. On the assumption that the implementation will compute the GHA only once, Table 3 summarizes the floating point operations required for each algorithm component and lists both the total operations required for each 3-tuple and the additional operations required for each scene center. Details of the complexity analysis for Sun azimuth and elevation, GCI sun vector, and GHA computations can be found in the white paper, *Sun Azimuth and Elevation Algorithms*, January 32, 1995.

Table C-3 Floating Point Operations for WRS Scene ID Algorithms

Algorithm	+/ -	–N		÷	MOD	Round	Trig
Longitude/ Latitude*	29	4	44	1	0	0	12
Azimuth/ Elevation	24	4	36	1	0	0	18
GCI Sun Vector	7	0	7	0	1	0	5
GHA	579	4	560	5	1	1	45
Total for each 3-tuple	608	8	604	6	1	1	67
Additional for each scene center	31	4	43	1	1	0	23

Assuming 250 scenes per day, 24 seconds per scene, and 4 PCD major frames per second, 58,964,000 + 12,005,750 N (N = the average number of floating point operations for a trig function) floating point operations per day will be required.

C.3 Estimated DSI

Table 4 presents DSI estimates for each of the algorithms. DSI for the top level scene identification algorithm are based on an assumed 5 DSI for each algorithm step plus a flat 100 DSI for initialization and maintenance of the WRS scene look-up table. DSI for the longitude/latitude computation is based on the number of floating point operations required (assuming 1 DSI per operation), the number of distinct intermediate values computed (assuming 1 DSI per value), a flat 100 DSI addition for matrix equation solution, a 30% overhead for initialization and termination handling, iteration, etc., and 100% overhead for exception handling. Details of the DSI estimates for Sun azimuth and elevation, GCI sun vector, and GHA computations appear in the white paper, *Sun Azimuth and Elevation Algorithms*, January 32, 1995.

* Estimates do not include cost of solving the vector equation in step 5 or of computing the cross scan and along mirror angles in step 4.

Table C-4 DSI Estimates for WRS Scene Identification Algorithms

Algorithm	Estimated DSI
WRS Scene Identification Top-Level	180
Latitude/Longitude Computation	433
Sun Azimuth/Elevation at Spacecraft	198
Sun Azimuth/Elevation at Earth	226
GCI Sun Vector	22
GHA	218
Total	1277

Appendix D - Acronym List

ACCA	Automatic Cloud Cover Assessment
ADP	Attitude Data Points
Ao	Operational Availability
ANSI	American National Standards Institute
AOS	Advanced Orbiting Systems
API	Applications Programming Interface
BCH	Bose-Chaudhuri-Hocquenghem (error detection and correction scheme)
BER	Bit Error Rate
CADU	Channel Access Data Unit
CASE	Computer Aided Software Engineering
CCA	Cloud Cover Assessment
CCB	Configuration Control Board
CIS	Centralized Information System
COTS	Commercial Off-the-Shelf
CPU	Central Processing Unit
CCSDS	Consultative Committee on Space Data System
CLCW	Command Link Control Word
CRC	Cyclic Redundancy Check
CRUD	Create, Retrieve, Update, Delete
CVCDU	Coded VCDU
DAMT	Distributed Application Monitor Tool
DAN	Data Availability Notice
DBMS	Database Management System
DD	Data Dictionary
DDE	Data Dictionary Entry
DDF	Data Distribution Facility
DDL	Data Definition Language
DFCB	Landsat & System, Data Format Control Book
DFD	Data Flow Diagram
DPCP	Distributed Process Control Program
DSI	Delivered Source Instruction
DSN	Deep Space Network
DSP	Digital Signal Processing
DTA	Data Transfer Acknowledgment
ECS	EOSDIS Core System
EDC	EROS Data Center
EDAC	Error Detection and Correction
EDP	Ephemeris Data Points
EOL	End of Line
EOSDIS	Earth Observation Data Information System

ER	Entity Relationship
ERD	Entity Relationship Diagram
EROS	Earth Resources Observation System
ESMO	Earth Science Mission Operations
ETM+	Enhanced Thematic Mapper Plus (instrument)
EPA	Euler Parameters
FDDI	Fiber Distributed Data Interface
FHS ERR	First Half Scan Error
FTAM	File Transfer Access and Management
FTP	File Transfer Protocol
F&PR	Functional and Performance Requirements
F&PS	Functional and Performance Specification
GByte	Gigabyte
GCI	Geocentric Inertial
GHA	Greenwich Hour Angle
GOTS	Government Off-the-Shelf
GSFC	Goddard Space Flight Center
GTSIM	Generic Telemetry Simulator
GUI	Graphical User Interface
HDF	Hierarchical Data Format
HDS	Horizontal Display Shift
HWC	Hardware Component
HWCI	Hardware Configuration Item
IAS	Image Assessment System
ICD	Interface Control Document
ID	Identification
IDD	Interface Data Description
IDPS	Image Data Processing Subsystem
IM	Information Modeling
IMU	Inertial Measurement Unit
IPD	Information Processing Division
ISO	International Organization for Standardization
Kbps	Kilobits per Second
LAN	Local Area Network
LCC	life-cycle cost
LDTS	LPS Data Transfer Subsystem
LGS	Landsat 7 Ground Station
LPS	Landsat 7 Data Processing System
LP DAAC	Land Processes Distributed Active Archive Center
LRU	Line Replaceable Unit
LZP	Level Zero Processing
L0R	Level Zero Reformatted

MACS	Management and Control Subsystem
Mbps	megabits per second
MFPS	Major Frame Processing Subsystem
MSCD	Mirror Scan Correction Data
MDT	Mean Downtime
MJF	Major Frame
MOC	Mission Operations Center
MO&DSD	Mission Operations and Data Systems Directorate
MTBF	mean time between failures
MTTR	mean time to repair
MTTRes	mean time to restore
NASA	National Aeronautics and Space Administration
NCC	Network Communication Center
NHB	NASA Handbook
NCSA	National Center for Supercomputing Applications
NMAS	Martin Marietta Astro Space
NMOS	Network Mission Operations Support
NOAA	National Oceanic and Atmospheric Administration
PCD	Payload Correction Data
PCDS	PCD Processing Subsystem
PN	Pseudo-Random Processing
QA	Quality Assurance
RAID	Redundant Array of Inexpensive Devices
RAM	Random Access Memory
RDCS	Raw Data Capture Subsystem
RDPS	Raw Data Processing Subsystem
RMA	Reliability, Maintainability, and Availability
RMS	Root, Mean, Square
RS	Reed-Solomon (error detection and correction scheme)
RT	Real Time
SCCS	Source Code Control System
SCLF	Search, Check, Lock, Flywheel
SCN DIR	Scan Direction
SD	System Design
SDL	Storage Definition Language
SDS	System Design Specification
SGI	Silicon Graphics, Incorporated
SHS ERR	Second Half Scan Error
SLDPF	Spacelab Data Processing Facility
SMP	Systems Management Policy
SN	Space Network
SQL	Structured Query Language

SRR	Software Requirements Review
SRS	Software Requirements Specification
SSDM	SEAS System Development Methodology
STDN	Spaceflight Tracking and Data Network
SV	Space Vehicle
SVR4	System V Release 4
SWCI	Software Configuration Item
TBD	To Be Defined/Determined
TBR	To Be Resolved
TMD	Telemetry Decommutation
UIL	User Interface Language
USGS	United States Geological Survey
UTC	Universal Time Coordinated
VCDU	Virtual Channel Data Unit
VCDU-ID	VCDU Identifier
VCID	Virtual Channel ID
VER	Version Number
VME	Versa Module European
WRS	World wide Reference System
WWV	Time Signal Radio Station with National Bureau of Standards information